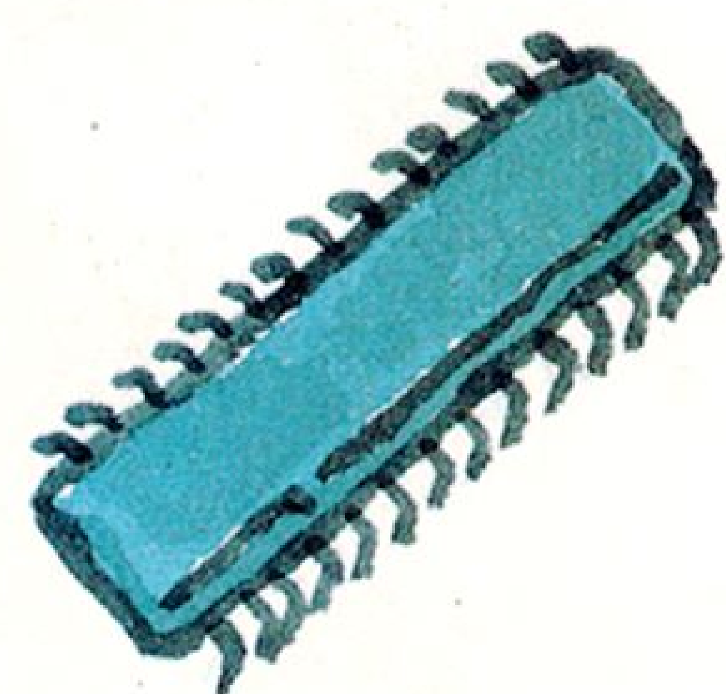
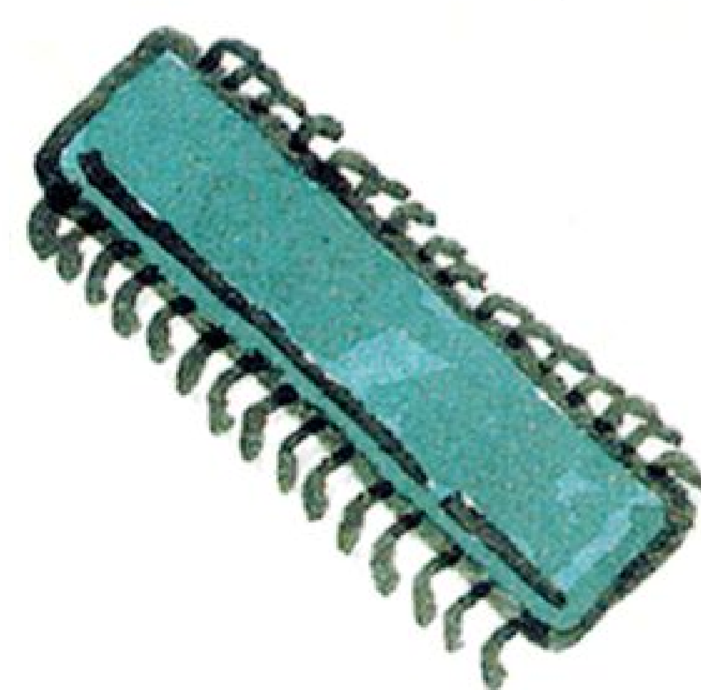
X1シリーズ
X1/turbo/Z

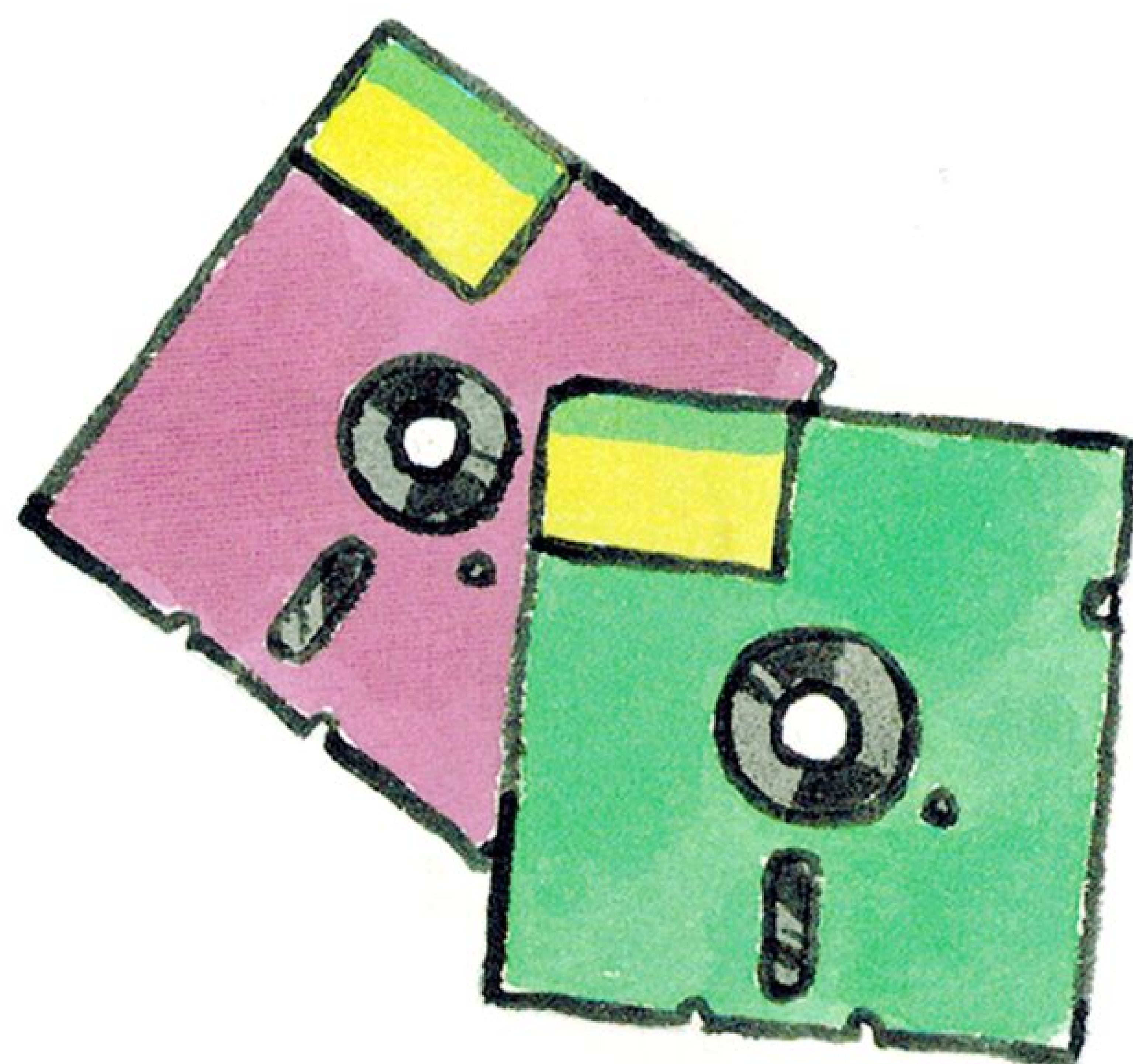
# 試験に出るX1

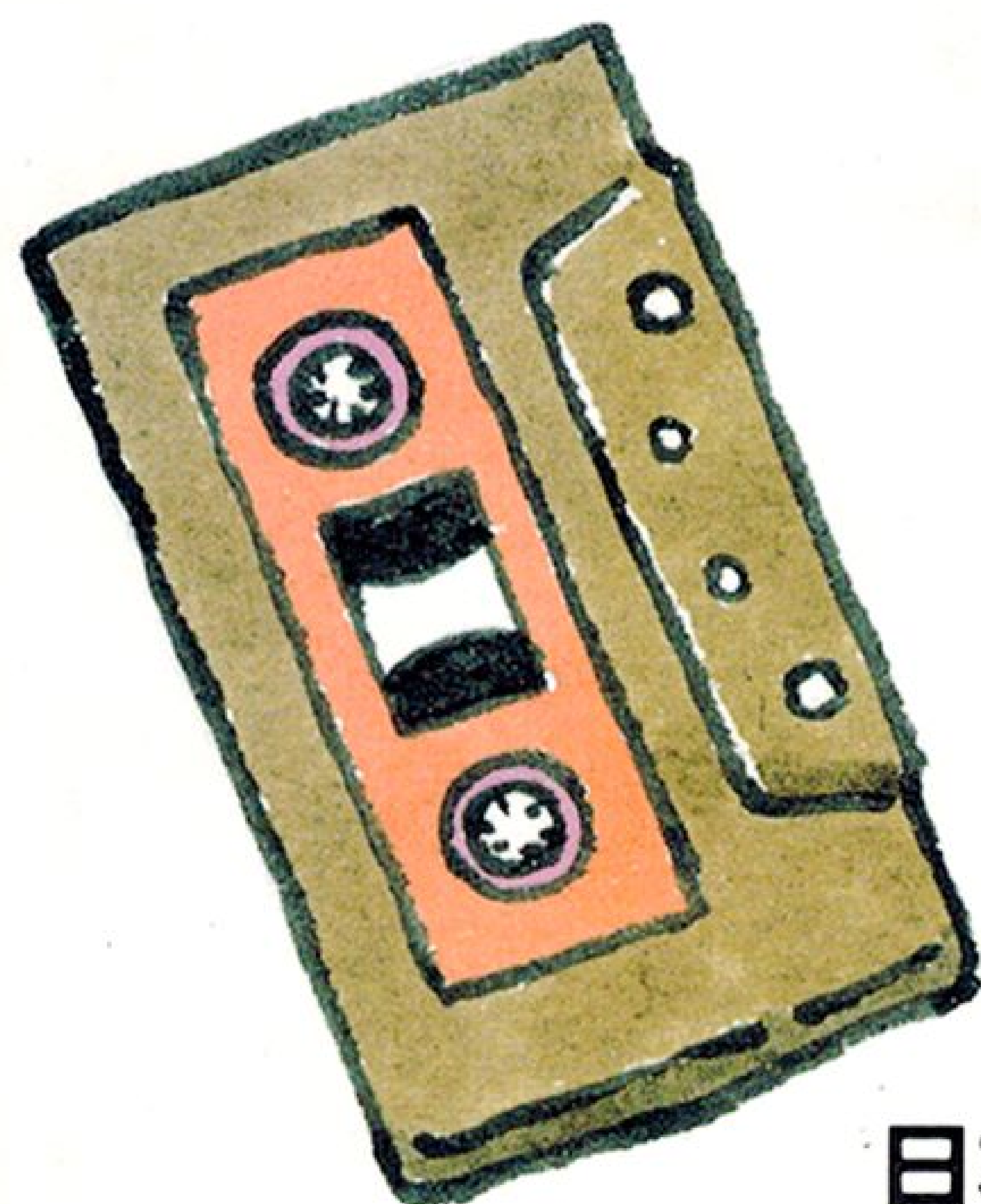
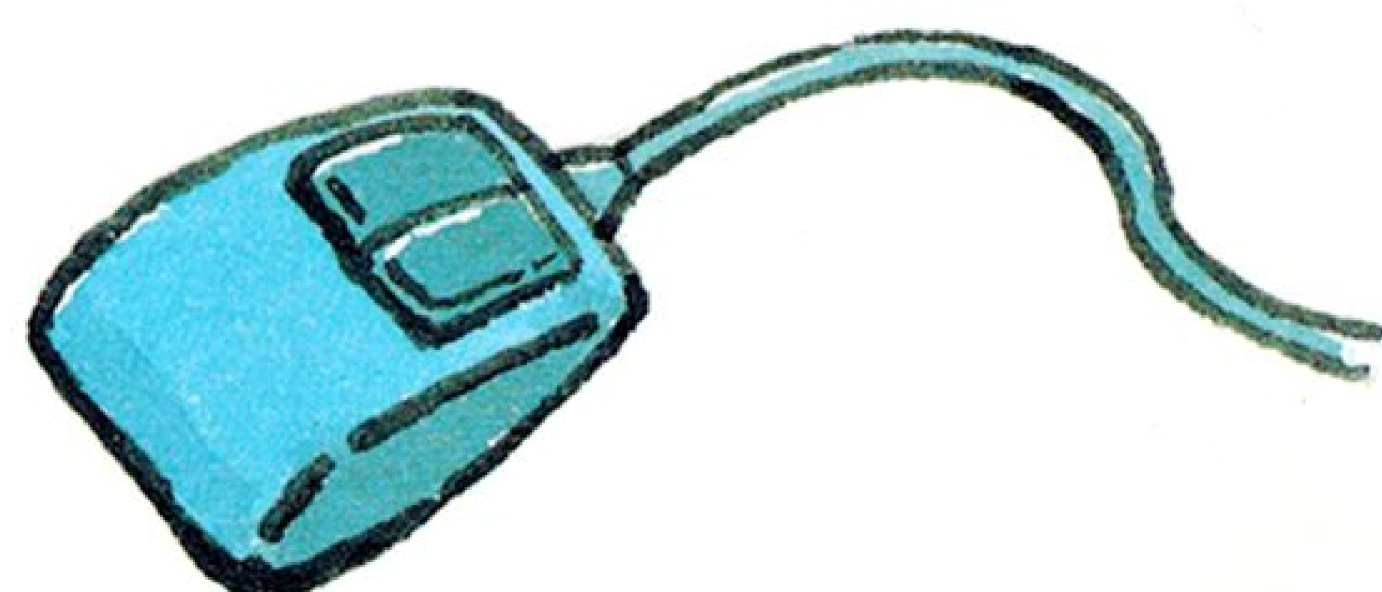
## ハードウェアのフルコース

祝　一平著

その筋なのは
美しき哉
一平



日本ソフトバンク

# 試験に出る

## ハードウェアのフルコース

祝　一平 著

日本ソフトバンク

# まえがき

'85年の6月からOh!MZ誌に連載を始め，それから2年2か月の間，X1のハードウェアをさまざまにいじくりまくった結果が本書である。ただし，連載時にあったライン描画，ペイントプログラムなどのグラフィックに関するものなどはページ数の関係で除外してある。世の中というものは，そんなものなのである。

連載を始めた当初は，分かっているつもりであったことも，実際に書き始めてみると意外といい加減な知識しかなかったことに気付き，あわてて調べ直したことも多々あった。それやこれやで，単純ミスからドタコまで，さまざまな間違いもあったが，そのたびに優秀な読者からご指摘をいただいた。そう考えてみると，多くのその筋なX1ユーザーのご助力を抜きにして，本書はなかったかもしれない。なんて，殊勝なことを言ってみたりする。

ともかく，本書にX1についての私の知識のほとんどを盛り込むことができた。読者各位に便利に使っていただきたい。そして，ブラックボックスではなく，すべてをユーザーに解放されたマシンの面白さを知っていただけたら幸いである。

私的ながら，本書は私にとって**1冊目**の著述にあたります。100冊を一つの頂点とするなら，ちょうどその1/100まで来たことになります。その（私にとって）記念すべき1冊にX1に巡り会えたことは，幸いなことだったと思います。

1987年　晩秋　　祝 一平

# 目　　次

# 目　次

# 目　次

# 目　次

# 第 0 章

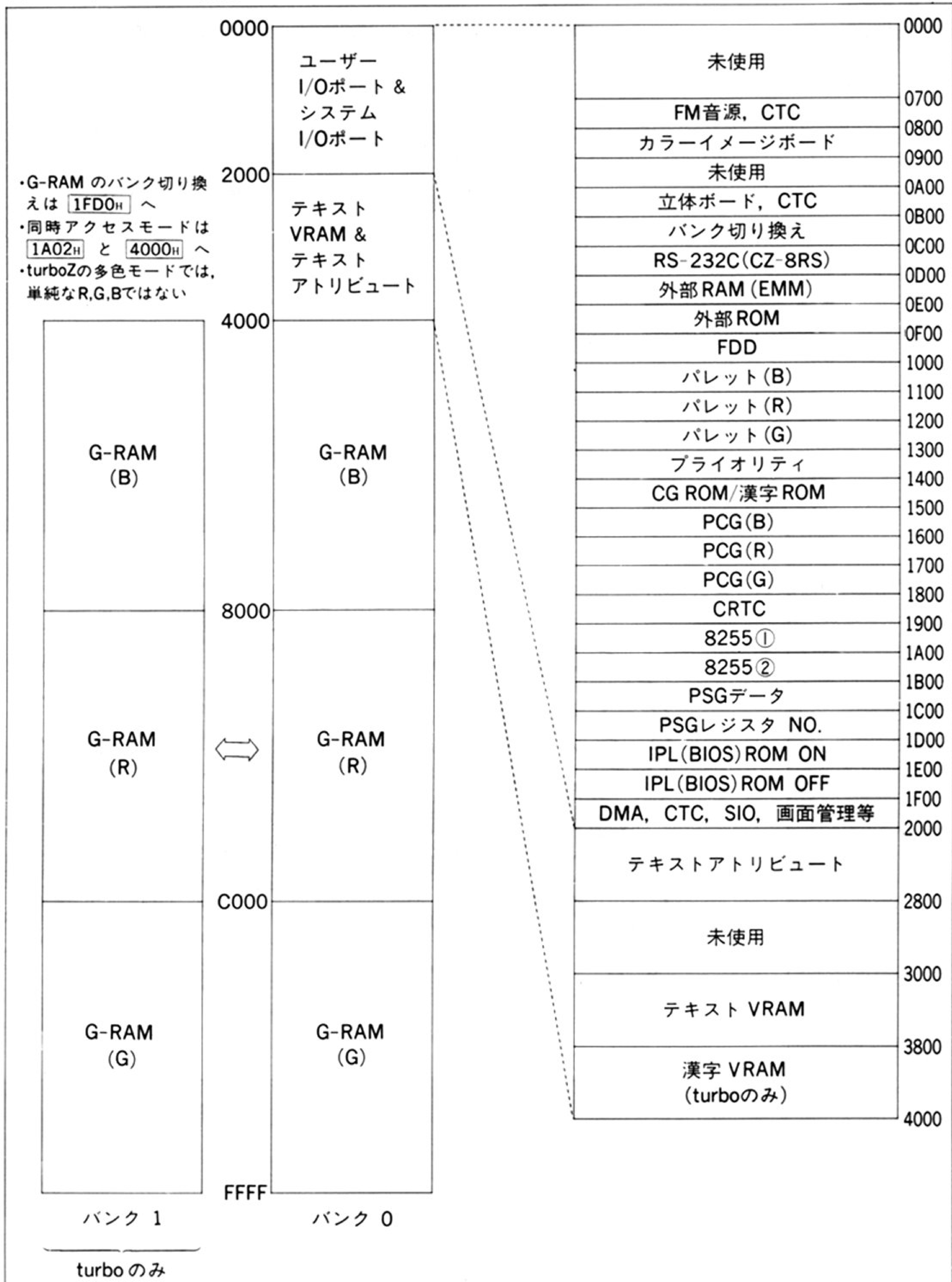
# I/Oマップ



きっと完全無欠なI/Oマップ

# 第0章
# きっと完全無欠なI/Oマップ・・・・

図 0-1　I/O マップの概観



この章ではX1/X1 turboのI/Oマップを網羅するのである。この部分はOh! MZ誌の連載時に，初めは「たぶん完全無欠なI/Oマップ」として書いたものであるが，あちこちに間違いが見つかり，やがて「ほとんど完全無欠なI/Oマップ」となり，さらにバグを取って，ここに「きっと完全無欠なI/Oマップ」となったのである。相変わらず完全無欠と言い切らないあたりが微妙にその筋なのである。

マップにはX1，turbo，Zの三つのマークがあるが，それらの意味は，

1） X1　　：ノンturboのユーザーにお勧め（もちろんturboでも使える）
2） turbo ：turboとZのユーザーにお勧め
3） Z　　 ：turboZのユーザーにだけお勧め

ということである。参考にしていただきたい。そして注意しておくが，X1のI/Oポートでは，たとえば$1802_H$番地は$1800_H$番地と同じである。これは業界用語で言うところの**デコードされてない**というやつである。

では，X1/X1 turboの64 KバイトにわたるI/O空間を見ていくのである。

まず，**図 0-1**はI/O空間の概観である。

$0000_H$～$1FFF_H$までは「ユーザーI/Oポート」と「システムI/Oポート」ということになっている。しかし，困ったことにどこまでが「ユーザー」でどこまでが「システム」なのかは，よく分からないのである。一応ユーザーI/O領域は$0000_H$～$00FF_H$,システムI/O領域は$0100_H$～$1FFF_H$と考えてよいだろう。

では，番地の小さい方から順に**もりもりと**やってみるのである。

## $0700_H$　FM音源/CTC（→第11章）
## IN/OUT：X1 / Z

| | | |
|---|---|---|
| $0700_H$ | YM2151アドレスポート | OUT |
| $0701_H$ | YM2151データポート | IN/OUT |
| $0704_H$ | CTCチャンネル0 | IN/OUT |
| $0705_H$ | CTCチャンネル1 | IN/OUT |
| $0706_H$ | CTCチャンネル2 | IN/OUT |
| $0707_H$ | CTCチャンネル3 | IN/OUT |

**チェックポイント**　turboZにはFM音源が入っているが，CTCは付いていない。ただし，$0704_H$はデータラッチ機能があり，FM音源機能のソフトチェックに使われる。

## $0800_H$ カラーイメージボード（→第12章）
## IN/OUT：X1

| | | |
|---|---|---|
| $0800_H$ | カラーイメージボードコントロール | OUT |
| $0801_H$ | 画像データ読み込み | IN |

**チェックポイント**　Zの画像デジタイズとはまったく違う。

## $0A00_H$　立体ボード/CTC
## IN/OUT：X1

| | | |
|---|---|---|
| $0A00_H$ | 立体ボードコントロール | OUT |
| $0A04_H$ | CTCチャンネル0 | IN/OUT |
| $0A05_H$ | CTCチャンネル1 | IN/OUT |
| $0A06_H$ | CTCチャンネル2 | IN/OUT |
| $0A07_H$ | CTCチャンネル3 | IN/OUT |

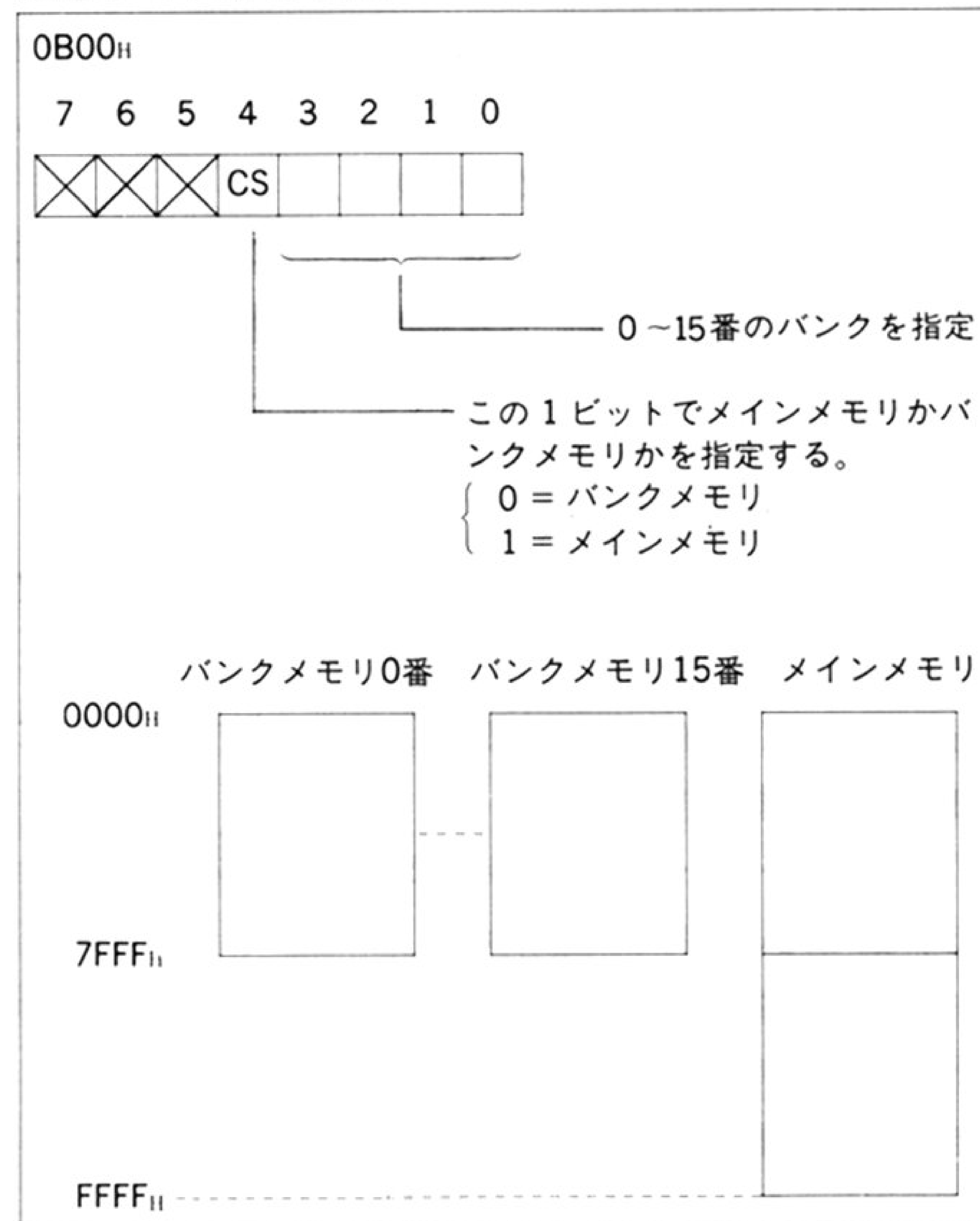
### コントロール内容

| コントロールデータ | $D_1$ $D_0$ | データ出力時のフィールドにおける立体スコープ動作 | 表示するグラフィックスクリーン |
|---|---|---|---|
| 0 | 0　0 | 垂直同期ごとL，R交互に開閉 | 垂直同期ごとL,R交互に表示 |
| 1 | 0　1 | 左シャッタOPEN　右シャッタCLOSE | LEFT(PAGE0) |
| 2 | 1　0 | 左シャッタCLOSE　右シャッタOPEN | RIGHT(PAGE1) |
| 3 | 1　1 | 左シャッタOPEN　右シャッタOPEN | |

注1）　$D_6$～$D_2$は無効。
注2）　シャッタのOPEN/CLOSEは一度設定すると，それ以後はハードがフィールドごとの切り換えを自動的に処理してくれる。詳しくはCZ-8BR1の取扱説明書の付録を参照のこと。

## 0B00H　メイン/バンクメモリ切り換え

OUT : turbo




turboで拡張された部分である。turboZ IIではバンクメモリ0，1番が内蔵されている。以前は，MS(X)-DOS(Multiplan)によってバンク0，1番が使われていただけであった。ノーマルな状態のバンクはシステム内メモリである。

## 0C00H　RS-232Cカード(CZ-8RS)

IN/OUT : X1

| | | |
|---|---|---|
| 0C＊0H | データ R/W | IN/OUT |
| 0C＊1H | コントロール，ステータス R/W | IN/OUT |
| 0C＊2H | 送信IEOをリセット | OUT |
| 0C＊3H | 受信IEOをリセット | OUT |
| 0C＊4H | 送信割り込み許可 | OUT |
| 0C＊5H | 送信割り込み禁止 | OUT |
| 0C＊6H | 受信割り込み許可 | OUT |
| 0C＊7H | 受信割り込み禁止 | OUT |

**チェックポイント**　アドレス中の"＊"はディップスイッチで設定する。

X1シリーズの旧タイプのRS-232Cカード（CZ-8RS）である。turboのモデル10以外で内蔵しているもの，およびRS-232Cマウスボード（CZ-8BM2）とは異なる。

## 0D00$_H$　外部RAMボード(EMM)(→第8章)
## IN/OUT : X1

EMM0の場合は次のようになる。

| | | |
|---|---|---|
| 0D00$_H$ | アドレス下位指定(00$_H$～FF$_H$) | OUT |
| 0D01$_H$ | アドレス中位指定(00$_H$～FF$_H$) | OUT |
| 0D02$_H$ | アドレス上位指定(00$_H$～04$_H$) | OUT |
| 0D03$_H$ | データのリード/ライト(内部アドレスは自動加算) | IN/OUT |

EMM1は0D04$_H$, EMM2は0D08$_H$から，というように4番地ごとに計64枚が（スロットの数があれば）つながる。

## 0E00$_H$　外部ROM(BASIC ROM, 漢字ROM)(→第3章)
## IN/OUT : X1

0E00$_H$～0E03$_H$ BASIC ROM (CZ-8RB)

| | | |
|---|---|---|
| 0E00$_H$ | アドレス上位指定 | OUT |
| 0E01$_H$ | アドレス中位指定 | OUT |
| 0E02$_H$ | アドレス下位指定 | OUT |
| 0E03$_H$ | データリード | IN |

0E80$_H$～0E82$_H$ 漢字ROM(CZ-8KR)

| | | |
|---|---|---|
| 0E80$_H$ | 左側データ/アドレス下位指定(00$_H$～FF$_H$) | IN/OUT |
| 0E81$_H$ | 右側データ/アドレス上位指定(00$_H$～FF$_H$) | IN/OUT |
| 0E82$_H$ | (0E82$_H$) ←01　漢字ROMセレクト<br>(0E82$_H$) ←00　増設用EPROMセレクト | OUT |

0E80$_H$～0E82$_H$ 増設用EPROM(カナ漢字変換ROM)

| | | |
|---|---|---|
| 0E80$_H$ | ROM1データ/アドレス下位指定(00$_H$～FF$_H$) | IN/OUT |
| 0E81$_H$ | ROM2データ/アドレス上位指定(00$_H$～FF$_H$) | IN/OUT |
| 0E82$_H$ | (0E82$_H$) ←00　増設用EPROMセレクト<br>(0E82$_H$) ←01　漢字ROMセレクト | OUT |

BASIC ROMには，アドレスの自動カウントアップ機構はないようである。また，アドレスの指定は上位，中位，下位の順で外部ROMボードとは異なっていることに注意していただきたい。漢字ROMのアクセス方法などに関しては第3章を見ていただきたい。増設用EPROMは，8K×2個がささるようになっている。HuWP（ハドソン）に付属していたカナ漢字変換ROM（I・Oデータ機器製：PIO-3055-01）である。しかしながら，X1G model 20などはそのようなROMがささる場所はなく，「辞書はディスク上に持つ」という状況である。すなわち過去の遺物である。

## 0FD0$_H$　ハードディスク
## IN/OUT : X1

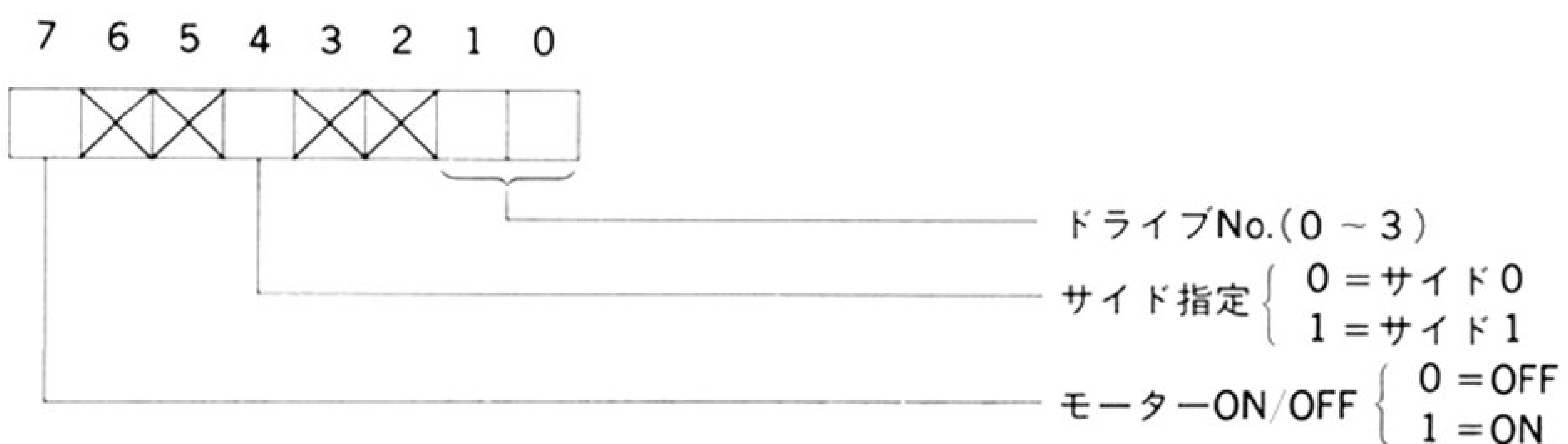
| 0FD0$_H$ | データ | IN/OUT |
|---|---|---|
| 0FD1$_H$<br>0FD2$_H$<br>0FD3$_H$ | コントロール | |

相変わらずアドレスしか分からない。ハードディスクはSCSI準拠であるから，将来的にはCD-ROMなども付くはずなのだが，どーなるのであろう。また，X1にもつながるはずなのであるが，'87年11月現在もサポートがない。むーん。

## 0FE8H 8インチFD
## IN/OUT : turbo

| | | |
|---|---|---|
| 0FE8H | ステータスレジスタ/コマンドレジスタ | IN/OUT |
| 0FE9H | トラックレジスタ | IN/OUT |
| 0FEAH | セクタレジスタ | IN/OUT |
| 0FEBH | データレジスタ | IN/OUT |
| 0FECH | FM方式指定/ドライブNo.，サイド，モーターON | IN/OUT |
| 0FEDH | MFM方式指定 | IN |
| 0FEEH | 1.6Mタイプ指定 | IN |
| 0FEFH | 500K・1M切り換え | IN |

0FECH出力内容の意味

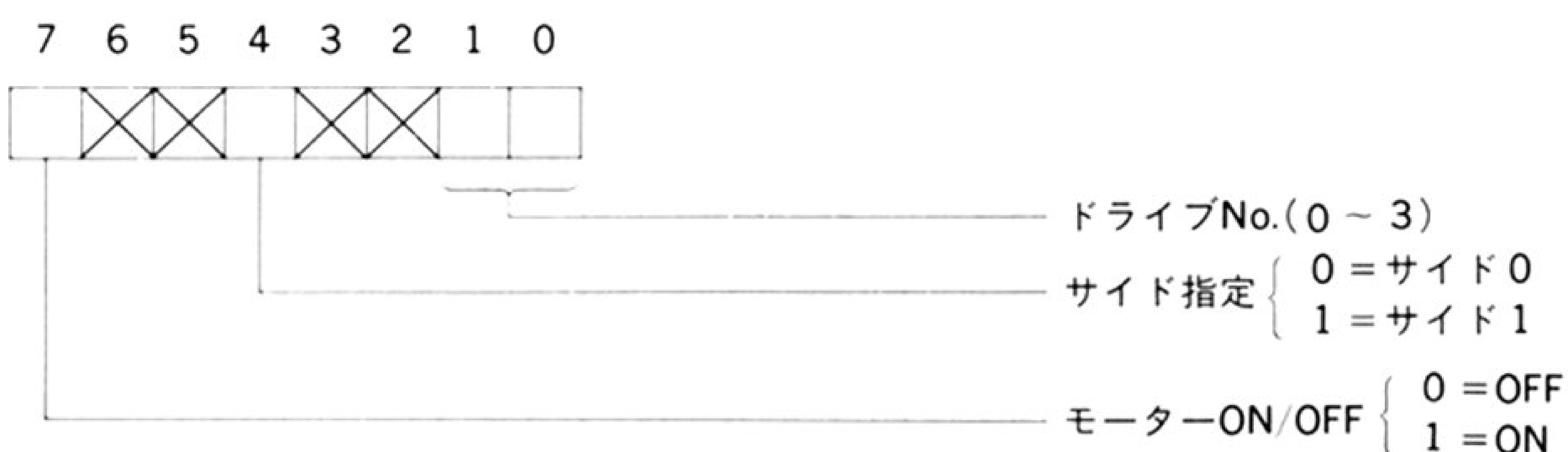



turbo/Zでは8インチFDを4台まで接続できるのである（5インチFDとは別）。しかし'87年11月現在もまったくサポートされていない。5インチFDの2HDが主流であるから，それも致し方ないであろう。8インチFDのコントロール内容は，I/Oアドレスがずれている以外は5インチ2HDと同じである。ただし，0FEFH番地は無意味である。

## 0FF8H 5インチFD（→第9章）
## IN/OUT : X1/turbo

| | | |
|---|---|---|
| 0FF8H | ステータスレジスタ コマンドレジスタ | IN/OUT |
| 0FF9H | トラックレジスタ | IN/OUT |
| 0FFAH | セクタレジスタ | IN/OUT |
| 0FFBH | データレジスタ | IN/OUT |
| 0FFCH | FM方式指定/ドライブNo.，サイド，モーターON | IN/OUT |
| 0FFDH | MFM方式指定 | IN |
| 0FFEH | 1.6Mタイプ(2HD)指定 | IN |
| 0FFFH | 500K(2D)・1M(2DD)切り換え | IN |

0FFCH出力内容の意味



## 1000$_H$, 1100$_H$, 1200$_H$　グラフィックパレット（→第14章）
## IN/OUT : X1 / Z

### X1/turboおよびZでのコンパチモード(OUTのみ)

この三つのI/Oポートは一組になっている。

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 10＊＊$_H$ | | | | | | | | | (00$_H$～FF$_H$) |
| 11＊＊$_H$ | | | | | | | | | (00$_H$～FF$_H$) |
| 12＊＊$_H$ | | | | | | | | | (00$_H$～FF$_H$) |

縦１列の３ビットを３桁の２進数(０～７)と
みなしカラーコードにする。

たとえば，第４ビットの縦１列が

| |
|---|
| 0 |
| 1 |
| 1 |

（右に90度回転すれば&B110＝6）

は「PALET 4, 6」に対応する。

**チェックポイント**　標準状態(「INIT」の後)は以下のとおりである。

(10＊＊$_H$) ←AA$_H$＝&B 1 0 1 0 1 0 1 0
(11＊＊$_H$) ←CC$_H$＝&B 1 1 0 0 1 1 0 0
(12＊＊$_H$) ←F0$_H$＝&B 1 1 1 1 0 0 0 0

「＊」印は何でもよいということである。またもや業界用語で言うところの「デコードされていない」である。このポートは出力のみである。うっかりINすると，パレットの状態が変化してしまう。注意していただきたい。

### turboZの多色モードの場合(IN/OUT可能)

**1)　4096色モード**

| | アドレス | | データ | |
|---|---|---|---|---|
| | G | R | B | B′ |
| 10 | 7 6 5 4 | 3 2 1 0 | 7 6 5 4 | 3 2 1 0 |
| | G | R | B | R′ |
| 11 | 7 6 5 4 | 3 2 1 0 | 7 6 5 4 | 3 2 1 0 |
| | G | R | B | G′ |
| 12 | 7 6 5 4 | 3 2 1 0 | 7 6 5 4 | 3 2 1 0 |

この12ビットでパレットコードを指定

この12ビット(4ビット×3回)でカラーコードを指定

**2)　64色モード**

上記のG，R，Bにおいて，各４ビット中の右側２ビットを00$_B$として指定する(G′，R′，B′はそのまま４ビットずつ)。

**チェックポイント**　Zでパレット R/Wには，1FC5$_H$も参照のこと。

## 1300$_H$ プライオリティ
### OUT：X1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

それぞれのカラーコードがテキストより優先するかを決定する。たとえば第5ビットが1ならば，カラーコード5(＝シアン)はテキストよりも優先する。第0ビットは，背景を意味する。
BASICのPRWに指定する数値と同じである。
**チェックポイント**　Zの場合は1FC0$_H$も参照。

## 1400$_H$, 1500$_H$, 1600$_H$, 1700$_H$, CG, 漢字ROM, PCGアクセス（→第2, 3章）
### IN/OUT：X1 / turbo

X1/X1 turboコンパチアクセスモード

| | | |
|---|---|---|
| 14＊＊$_H$ | CG ROMアクセス | IN |
| 15＊＊$_H$ | PCG BLUEアクセス | IN/OUT |
| 16＊＊$_H$ | PCG RED アクセス | IN/OUT |
| 17＊＊$_H$ | PCG GREENアクセス | IN/OUT |

X1 turbo高速アクセスモード

| | | |
|---|---|---|
| 14＊0$_H$～14＊F$_H$ | CG，漢字ROMアクセス | IN |
| 15＊0$_H$～15＊F$_H$ | PCG BLUEアクセス | IN/OUT |
| 16＊0$_H$～16＊F$_H$ | PCG REDアクセス | IN/OUT |
| 17＊0$_H$～17＊F$_H$ | PCG GREENアクセス | IN/OUT |

**チェックポイント**　turboの場合，I/Oポート1FD0$_H$の第5ビットが1ならば高速アクセスモード，0ならばコンパチアクセスモード。

## 1800$_H$ CRTC（→第1章）
### OUT：X1

| | |
|---|---|
| 1800$_H$ | CRTCレジスタNo.指定(0～17) |
| 1801$_H$ | CRTCレジスタへのデータ(00$_H$～FF$_H$) |

(CRTCの型番はHD46505-SP)
**チェックポイント** 40/80桁の切り換えには，I/Oポート1A02$_H$の第6ビットが関係している。0なら80桁，1なら40桁。
turboでは400ラインになったことに加えて，アンダーラインモードも加わったので，1FD0$_H$もいっしょに設定しなければならない場合がある。

18個あるCRTCのレジスタのうち実際に意味があるのは12個である。X1ではスーパーインポーズ時にスクロールできるが，これはCRTCの5番レジスタ（R5)を書き換え，さらにポート1A02$_H$の第4ビットを0にすることで行なえる。いろいろいじくりまわすと面白い石である。

## 1900$_H$ サブCPU80C49(8255①)(→第4章)
## IN/OUT : X1

| グループ | ポート端子 | コントロール内容 | アクティブ | |
|---|---|---|---|---|
| A | $PA_7$ | Z80とのデータ入出力(IN/OUT)(1900$_H$) | − | モード2 |
| | $PA_6$ | | − | |
| | $PA_5$ | | − | |
| | $PA_4$ | | − | |
| | $PA_3$ | | − | |
| | $PA_2$ | | − | |
| | $PA_1$ | | − | |
| | $PA_0$ | | − | |
| | $PC_7$ | Z80Aに対してデータ受け取り指示信号 | L | |
| | $PC_6$ | Z80Aがポート Aからデータ受け取り信号 | L | |
| | $PC_5$ | Z80Aに対してデータ転送禁止信号 | H | |
| | $PC_4$ | Z80Aからのデータをポート Aに入力/ラッチ指示信号 | L | |
| B | $PC_3$ | | − | モード0 |
| | $PC_2$ | カセットLEDの点灯(H:READ, L:WRITE) | − | |
| | $PC_1$ | Z80AへのBREAK信号 | L | |
| | $PC_0$ | カセットのEJECTソレノイドコントロール | L | |
| | $PB_7$ | OBF信号 | − | |
| | $PB_6$ | ACK信号 | − | |
| | $PB_5$ | APSS(無記録部検出) | − | |
| | $PB_4$ | EJECT SWセンス | L | |
| | $PB_3$ | | − | |
| | $PB_2$ | カセットテープの書き込み禁止用の爪がある状態 | H | |
| | $PB_1$ | カセットがセットされている状態 | H | |
| | $PB_0$ | テープエンド検出 | L | |

**チェックポイント**　この図は80C49から見た場合である。このときZ80は80C49の周辺デバイスとみなされる。Z80から入出力が可能なのはポートA($PA_0$〜$PA_7$)だけである。

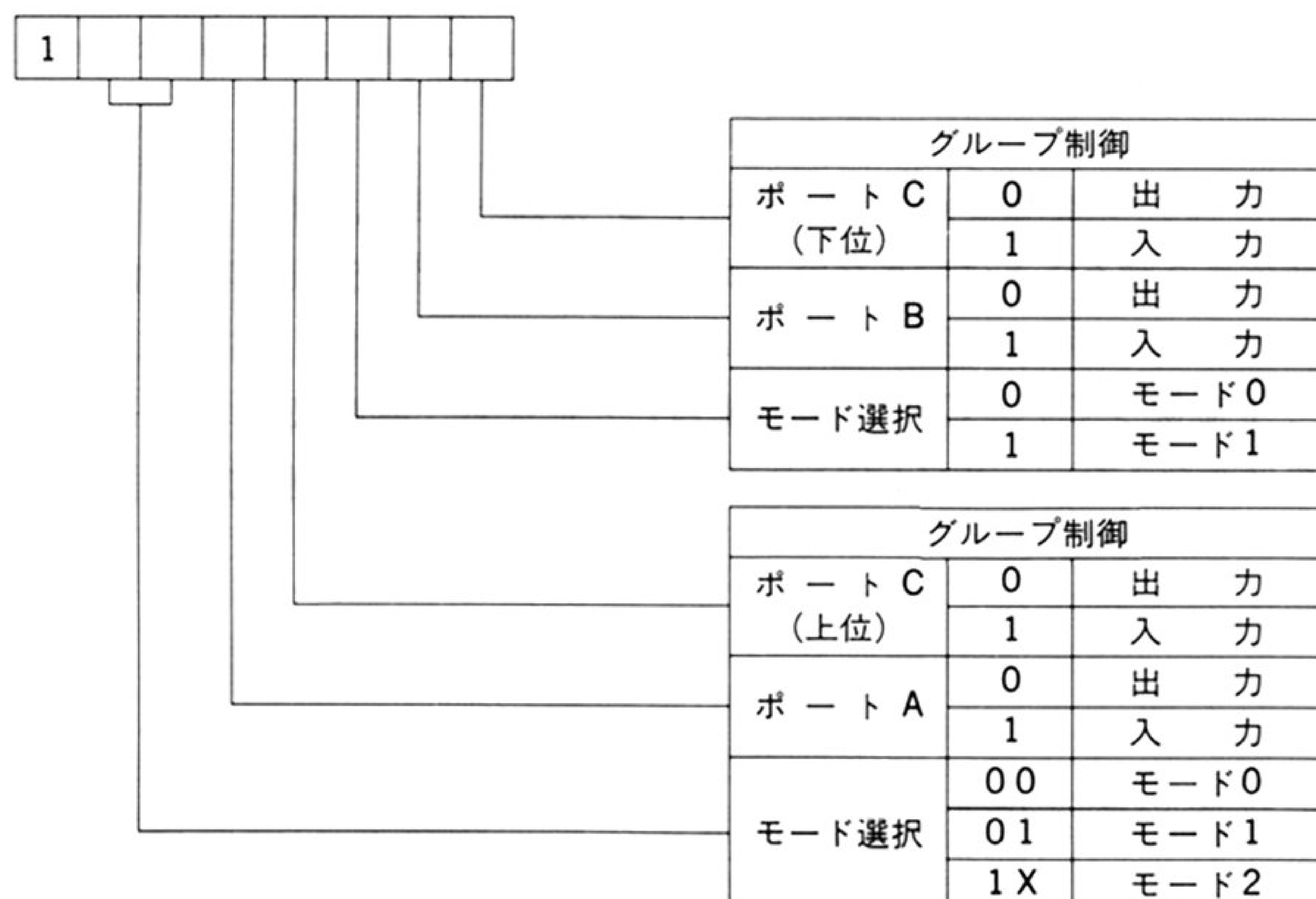
チェックポイントを繰り返すが，ポートB ($PB_0$〜$PB_7$)，ポートC ($PC_0$〜$PC_7$) をZ80が直接変化させることはできない。サブCPU 80C49と交信して間接的にアクセスしなければならない。結局，実際に意味があるのは1900$_H$番地だけということであり，Z80はこのポートを使ってサブCPUと会話することになる。当然8255のモード設定などはZ80側からは行なえない。サブCPUのアクセスには8255②を参照のこと。

## 1A*$0_H$〜1A*$3_H$　8255②

## IN/OUT : X1

| ポート | ポート端子 | コントロール内容 | アクティブ | |
|---|---|---|---|---|
| A OUT ($1A00_H$) | $PA_7$ | プリンタ出力データ | − | モード0（グループA） |
| | $PA_6$ | | − | |
| | $PA_5$ | | − | |
| | $PA_4$ | | − | |
| | $PA_3$ | | − | |
| | $PA_2$ | | − | |
| | $PA_1$ | | − | |
| | $PA_0$ | | − | |
| B IN ($1A01_H$) | $PB_7$ | 垂直帰線期間信号 | L | モード0（グループB） |
| | $PB_6$ | データ転送禁止信号 | H | |
| | $PB_5$ | 80C49からのデータ受け取り可能指示信号 | L | |
| | $PB_4$ | BIOS ROM ON(turboのみ) | L | |
| | $PB_3$ | プリンタからの入力可能指示信号 | L | |
| | $PB_2$ | 垂直同期信号 | H | |
| | $PB_1$ | カセット読み出しデータ | − | |
| | $PB_0$ | BREAK信号 | L | |
| C OUT ($1A02_H$) | $PC_7$ | 立ち上がりでプリンタは入力データをサンプルする | 立ち上げ | |
| | $PC_6$ | 80/40桁（1=40桁，0=80桁） | − | |
| | $PC_5$ | 立ち下げで同時アクセスモード | 立ち下げ | |
| | $PC_4$ | スムーススクロール信号 | L | |
| | $PC_3$ | | − | モード0（グループA） |
| | $PC_2$ | | − | |
| | $PC_1$ | | − | |
| | $PC_0$ | カセットテープへの書き込みデータ | − | |
| コントロールレジスタ設定ポート　OUT ($1A03_H$) | | | | |

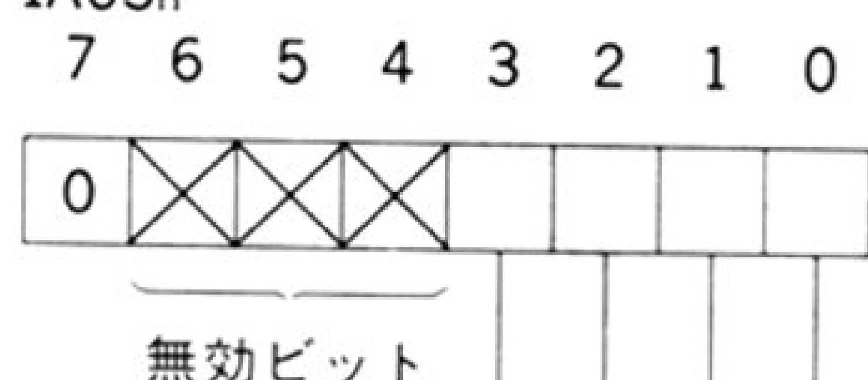
8255モード制御（ビット7=1）
$1A03_H$




| グループ制御 | | |
|---|---|---|
| ポートC（下位） | 0 | 出力 |
| | 1 | 入力 |
| ポートB | 0 | 出力 |
| | 1 | 入力 |
| モード選択 | 0 | モード0 |
| | 1 | モード1 |

| グループ制御 | | |
|---|---|---|
| ポートC（上位） | 0 | 出力 |
| | 1 | 入力 |
| ポートA | 0 | 出力 |
| | 1 | 入力 |
| モード選択 | 00 | モード0 |
| | 01 | モード1 |
| | 1X | モード2 |

8255ビット・セット/リセット(ビット7=0)(ポートCに対して)
1A03H




| ビット・セット/リセット | | |
|---|---|---|
| | 0 | リセット |
| | 1 | セット |

| ポートC　ビット選択 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ビット | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| $D_1$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $D_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $D_3$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**チェックポイント**　ビット・セット/リセット制御はポートC(1A02H)の任意の1ビットを変化させるもので，プログラムを高速かつコンパクトにできる。

8255②は①と違って別に変わった使われ方はしていない。ここは重要なので少々丁寧に説明する。

プリンタへの出力は，

1）BUSY（$PB_3$）=0まで待つ

2）ポートA（1A00H）へデータを出力

3）$PC_7$を立ち上げる（?→0→1）

で行なえる。

$PB_7$（垂直帰線期間信号），$PB_2$（垂直同期信号）はCRTの状態読み出し。特に$PB_7$はPCGアクセス時に重要である。

$PB_6$（データ転送禁止信号）はサブCPUとの会話用に使われていて，$PB_6$=1はサブCPUからの「待ってくれ」を意味する。

$PB_4$は現在のバンクがBIOS（IPL）ROM側か，メインメモリ側かを検出するものである。turboのみの機能である。

$PB_1$はカセットからの読み出しデータ。

$PB_0$はサブCPUからの信号で，CMTがPLAY中に[BREAK]キーが押されたときなどに0になる。

$PC_6$は80，40桁の切り換え。

$PC_5$は立ち下げ（?→1→0）で同時アクセスモードになる。ただし，その前にDIで割り込みを禁止しておくこと。このモードを解除するには，何でもいいからIN命令を実行すればよい。詳しくは[4000H]へ。

$PC_4$は0のときCRTCの5番レジスタとともにスムーススクロールを実現する（スーパーインポーズ時）。

$PC_0$はカセットテープへの書き込みであるが，任意の長さの「1」を書けるわけではない。

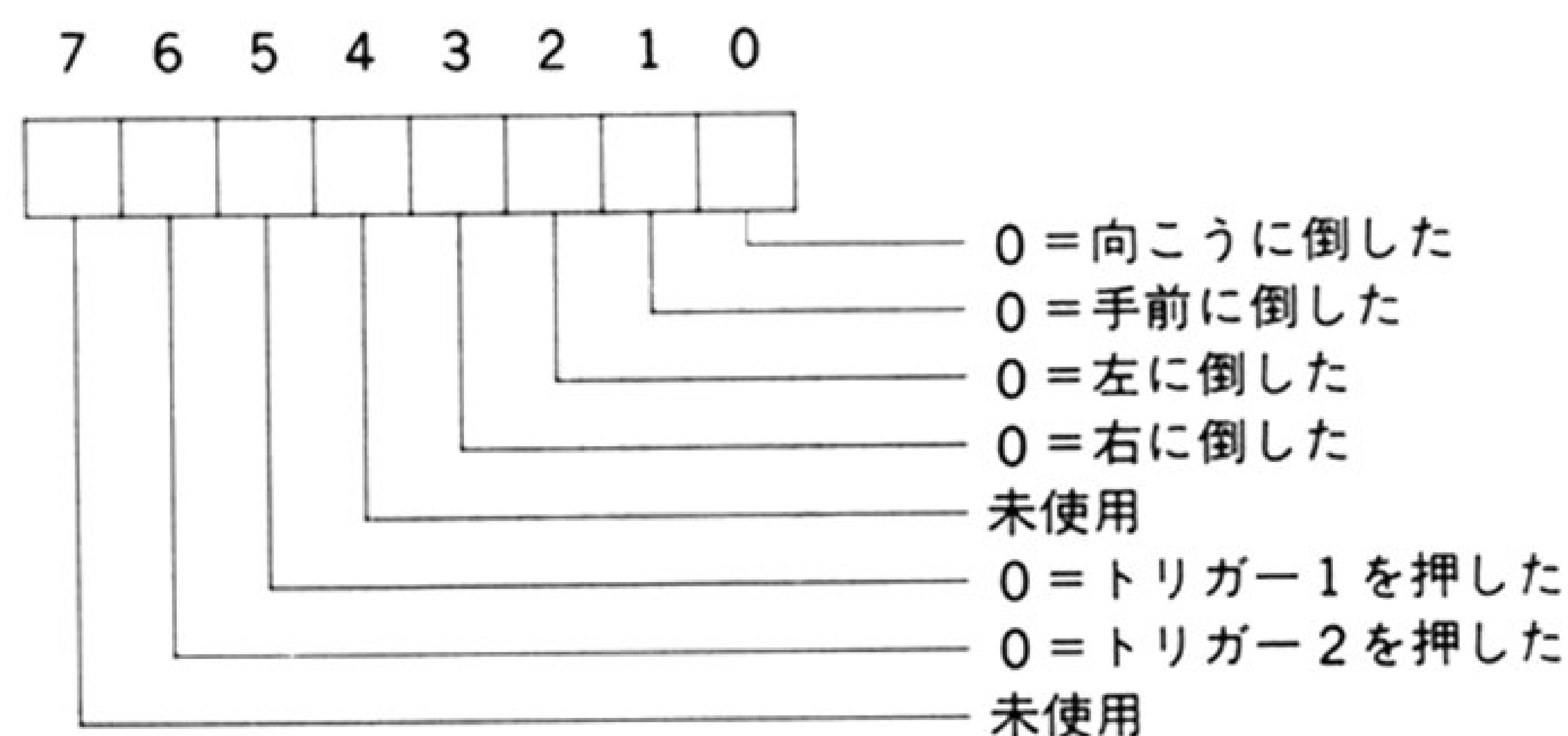
なお，モード設定はIPLが行なってくれるので別にユーザーは気にする必要はない。

## 1B＊＊H, 1C＊＊H PSG, ジョイスティック（→第10章）

IN/OUT：X1

| | | |
|---|---|---|
| 1B＊＊H | PSGデータ(00H～FFH) | IN/OUT |
| 1C＊＊H | PSGレジスタ指定(0～15) | OUT |

ジョイスティックのデータの意味($R_{14}$, $R_{15}$)




PSG（AY-3-8910）へのアクセスである。ジョイスティックをアクセスするにはPSGのレジスタ，$R_7$，$R_{14}$，$R_{15}$を使う。次のプログラムは，JOY1，JOY2をともに読み出し，データを表示するものである。

```
10 OUT &H1C00, 7：'select register
20 r5=INP(&H1B00) AND &B111111
30 OUT&H1C00, r5：'set register
40 '
50 OUT &H1C00, 14：j1=INP(&H1B00)
60 OUT &H1C00, 15：j2=INP(&H1B00)
70 PRINT BIN$ (j1), BIN$ (j2)
80 GOTO 50
```

ジョイスティックからの入力は負論理である。なお，レジスタ番号の指定は1度行なえば続けてアクセスする際には再指定する必要はない。

## 1D＊＊H, 1E＊＊H IPL(BIOS)ROM ON/OFF

OUT：X1 / turbo

| | | |
|---|---|---|
| 1D＊＊H | IPL ROM ON | OUT |
| 1E＊＊H | IPL ROM OFF | OUT |

**チェックポイント** OUTするデータはなんでもよい。

ROMをONにすると0000H～7FFFHまでがROMに切り換わる。したがって，それを直接行なうOUT命令は8000H～の位置になければならない。

ROMがONのとき0番地へジャンプすればIPLが起動する。ROMがONのときに，0000H～7FFFHにデータの書き込みを行なうと，RAMに書き込まれる。しかし読み出しはROMから。この手法はシャドウRAMと呼ばれる。これにより64Kバイトを一発でロードできる。

1A01H番地の$PB_4$も参照。

```
LD      BC, 1D00H
OUT     (C), A
XOR     A
LD      (100H), A
LD      A, (100H)
LD      BC, 1E00H
OUT     (C), A
```

とすると，Aレジスタに何が入っているか？　0ではない。

### 1F8*H　DMA（→第8章）
### IN/OUT：turbo

| | | |
|---|---|---|
| 1F8*H | DMAへのコマンド，データ | IN/OUT |

Z80 DMAコントローラは(メモリ，I/O)↔(メモリ，I/O)間のデータ転送を高速に行なうためのLSIである。DMAにはデータ転送ばかりではなくサーチ機能もある。また転送でも特定の番地の内容を，ある範囲にコピーすることができる。すなわちG-RAM全域に0を転送することも可能である。この手法を使えば高速画面クリア，スクロールを実現できるが，turbo BASICでは使用していないようである。

### 1F90H　SIO（→第6,7章）
### IN/OUT：X1/turbo

| | | |
|---|---|---|
| 1F90H | チャンネルAデータ | IN/OUT |
| 1F91H | チャンネルA制御 | IN/OUT |
| 1F92H | チャンネルBデータ | IN/OUT |
| 1F93H | チャンネルB制御 | IN/OUT |

**チェックポイント**　チャンネルBはマウスにつながっている。ボーレートは4800bpsである。

RS-232Cカード，CZ-8RSとは互換性がないことに注意が必要。

CZ-8BM2上のSIOのアドレスは$1F98_H$〜$1F9B_H$が割り当てられている。このカードにはCTCが入っていて，こちらの方のアドレスは$1FA8_H$〜$1FAB_H$となる。これはCTCがボーレートジェネレータの役割をしているためである。

### 1FA0H　CTC（→第5章）
### IN/OUT：X1/turbo

| | | | |
|---|---|---|---|
| 1FA0H | チャンネル0 | タイマモード | IN/OUT |
| 1FA1H | チャンネル1 | SIOチャンネルAクロック | IN/OUT |
| 1FA2H | チャンネル2 | SIOチャンネルB(マウス)クロック | IN/OUT |
| 1FA3H | チャンネル3 | カウンタモード | IN/OUT |

**チェックポイント**　チャンネル0の使うクロックは4MHz，タイマ周期は4μsec〜16.384msecまで。チャンネル1，2は2MHzのクロックを使用。チャンネル3はチャンネル0をカウントし，最長タイマは4.194sec。

SIOの所で説明したように，1F98H～1F9BHにもう一つのCTCを付けることができる。また，FM音源ボード，立体ボードなどにもCTCが載る。

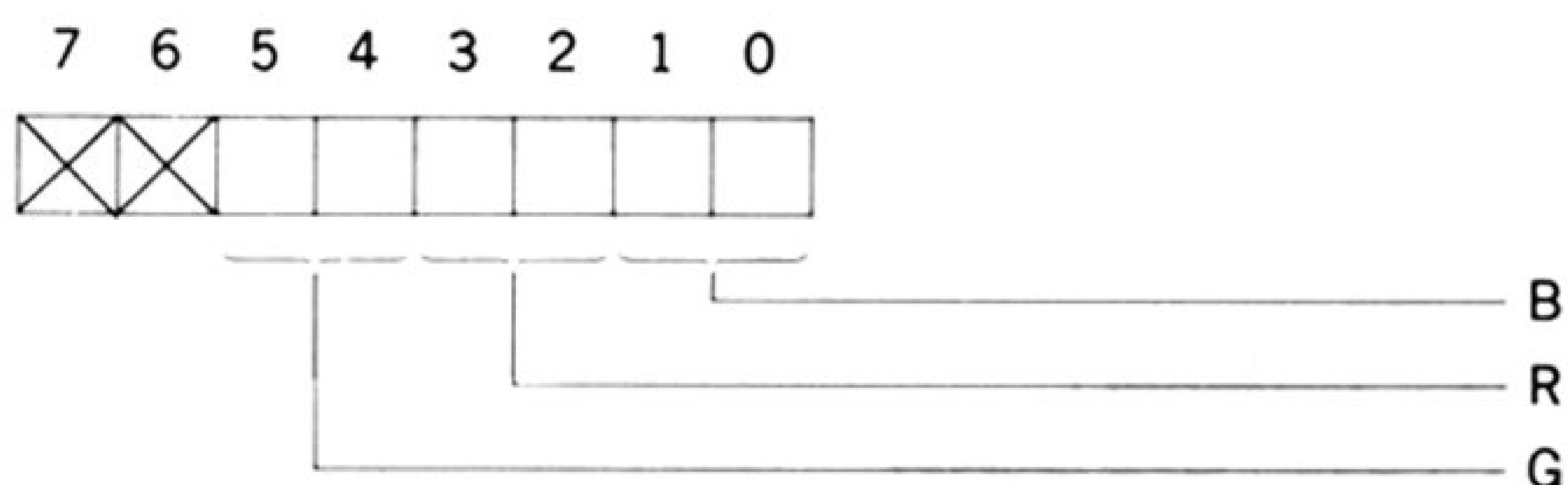
### 1FB0H　Zモード指定（→第14章）
### IN/OUT：Z

| データ内容 | コントロール |
|---|---|
| ビット0 | 0＝インタレーススーパーインポーズしない<br>1＝インタレーススーパーインポーズする |
| ビット1 | 無効 |
| ビット2 | 0＝画像取り込みの階調ノーマル<br>1＝画像取り込みの階調反転　（ビット7，3＝1のときのみ有効） |
| ビット3 | 0＝画像取り込みをしない<br>1＝画像取り込みをする　（ビット7＝1のときのみ有効） |
| ビット4 | 0＝4096色1画面モード指定<br>1＝64色2画面モード指定　（320×200のときのみ有効） |
| ビット5 | 無効 |
| ビット6 | 無効 |
| ビット7 | 0＝X1/X1 turboコンパチモード<br>1＝多色(turboZ)モード |

### 1FB9H～1FBFH　Zテキストパレット指定（→第14章）
### IN/OUT：Z

| アドレス | コントロール |
|---|---|
| 1FB9H | 青のカラーコード |
| 1FBAH | 赤のカラーコード |
| 1FBBH | マゼンタのカラーコード |
| 1FBCH | 緑のカラーコード |
| 1FBDH | シアンのカラーコード |
| 1FBEH | 黄のカラーコード |
| 1FBFH | 白のカラーコード |

設定データ




RGB各2ビットずつ＝64色を指定できる。

### 1FC0H　Zプライオリティ指定（→第14章）
### IN/OUT：Z

| データ内容 | コントロール |
|---|---|
| ビット0，1 | 0，0＝テキストはグラフィックより優先<br>0，1＝グラフィックはテキストより優先<br>1，0＝テキストはグラフィック2面の間に入る<br>1，1＝未定義 |

| | |
|---|---|
| ビット2 | 無効 |
| ビット3 | 0＝バンク0はバンク1より優先<br>1＝バンク1はバンク0より優先 |
| ビット4 | 0＝片方のバンクだけを表示する<br>1＝バンク0, 1を同時に表示する |
| ビット5<br>ビット6<br>ビット7 | 無効 |

ビット3, 4は2画面モード(→1FB0H番地のビット4)のときのみ有効。

- 多色(turboZ)かつ320×200モードでのみ意味のあるポート
- ビット4＝0のときはビット1＝0とみなされる
- ビット4＝1のときは1FD0Hのビット3は無効

## 1FC1H　Z画像取り込み位置補正指定（→第14章）

## IN/OUT：Z

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

0～255の補正ドット数を指定する。

- 200ラインモード(1FD0Hのビット0＝0)のときのみ有効

## 1FC2H　Zモザイク/量子化取り込み指定（→第14章）

## IN/OUT：Z

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

X方向モザイク

| ビット2 | ビット1 | ビット0 | ドット数 |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | 64 |
| 1 | 1 | 1 | − |

Y方向モザイク

| ビット5 | ビット4 | ビット3 | ドット数 |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | − |
| 1 | 1 | 1 | − |

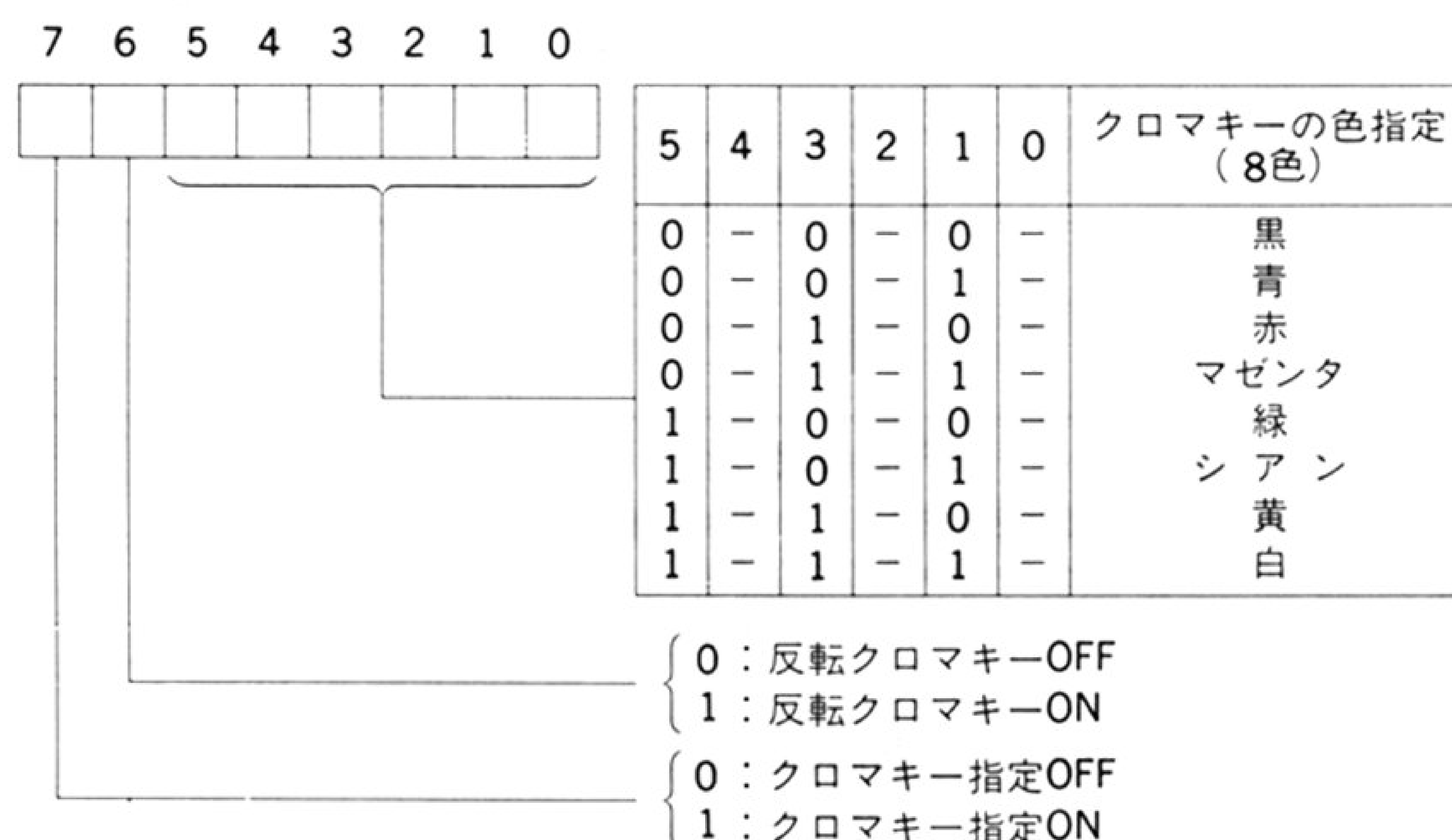
量子化指定

| ビット7 | ビット6 | 階調 |
|---|---|---|
| 0 | 0 | 4ビット階調取り込み(4096色) |
| 0 | 1 | 3ビット階調取り込み(512色) |
| 1 | 0 | 2ビット階調取り込み(64色) |
| 1 | 1 | 1ビット階調取り込み(8色) |

**チェックポイント**　64色モードを指定してある場合はビット7＝1として扱われる。

## 1FC3H　Zクロマキー指定（→第14章）

IN/OUT：Z




| 5 | 4 | 3 | 2 | 1 | 0 | クロマキーの色指定（8色） |
|---|---|---|---|---|---|---|
| 0 | − | 0 | − | 0 | − | 黒 |
| 0 | − | 0 | − | 1 | − | 青 |
| 0 | − | 1 | − | 0 | − | 赤 |
| 0 | − | 1 | − | 1 | − | マゼンタ |
| 1 | − | 0 | − | 0 | − | 緑 |
| 1 | − | 0 | − | 1 | − | シアン |
| 1 | − | 1 | − | 0 | − | 黄 |
| 1 | − | 1 | − | 1 | − | 白 |

ビット0，2，4は抜けている。
ビット7＝0のときビット6は無効。
クロマキーとは，映像画面中の指定した色を抜いて，そこにコンピュータ画面をはめ込むものである。

## 1FC4H　Zスクロール指定（→第14章）

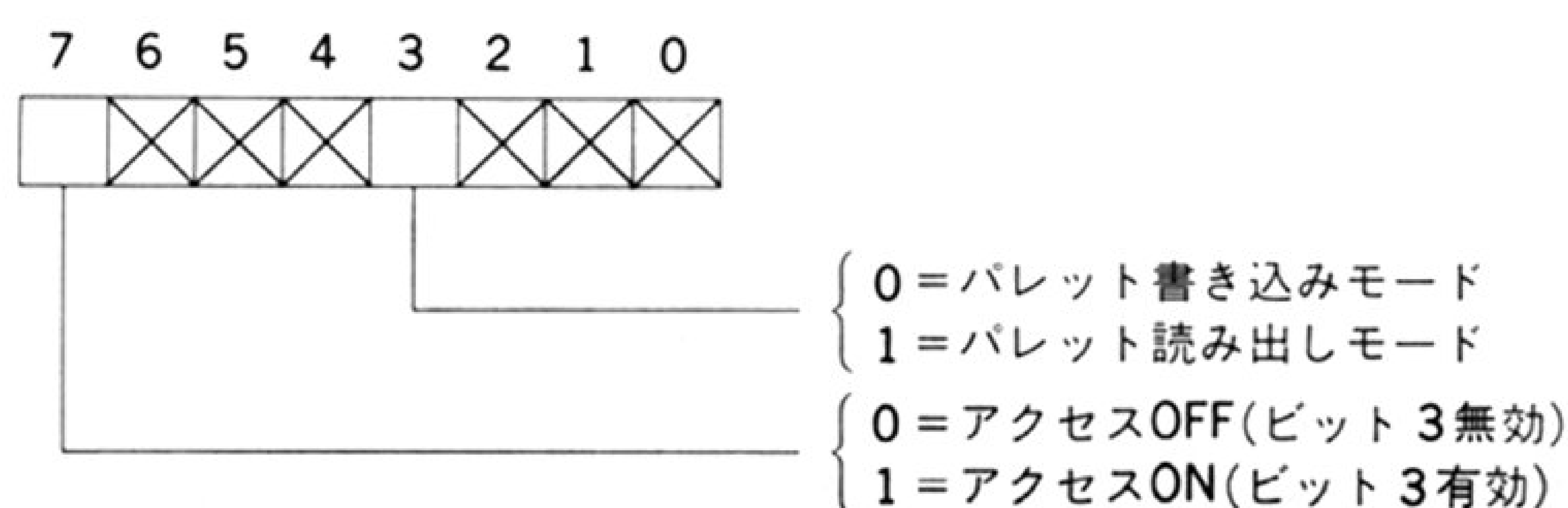
IN/OUT：Z

| データ内容 | コントロール |
|---|---|
| ビット0 | 0＝スクロールインする<br>1＝スクロールアウトする |
| ビット1 | 0＝スクロールイン，アウトを繰り返す(旧モード)<br>1＝一度出たら，スクロールインしない |
| ビット2 | 0＝CRT出力する<br>1＝CRT出力しない |
| ビット3 | 0＝ビット0～2を無効とする<br>1＝ビット0～2を有効とする |
| ビット4<br>～<br>ビット7 | 未使用 |

**チェックポイント**　ビット3＝0のときビット0～2は無効。これは，スーパーインポーズと組み合わせて，「一度だけスクロール」を実現するものである。CRTCと8255②も必要。

## 1FC5H　Z多色モードでのグラフィックパレット制御指定（→第14章）

IN/OUT：Z




**チェックポイント**　このポートは多色モードでのみ有効。10**H～12**Hと組み合わせて使う。

## 1FD0H 画面管理（→第1章）
## (IN)/OUT：turbo

| データ内容 | コントロール |
|---|---|
| ビット0 | 0＝低解像度モニタ(200ライン)<br>1＝高解像度モニタ(400ライン) モニタ切り換え |
| ビット1 | 0＝1本ラスタ/ドット<br>1＝2本ラスタ/ドット |
| ビット2 | 0＝ノーマル(8ラスタ/CHAR)(25行，20行)<br>1＝漢字　(16ラスタ/CHAR)(12行，10行) |
| ビット3 | 0＝バンク0表示<br>1＝バンク1表示 |
| ビット4 | 0＝バンク0アクセス<br>1＝バンク1アクセス |
| ビット5 | 0＝PCGコンパチアクセス<br>1＝PCG高速アクセス |
| ビット6 | 0＝8ラスタCGアクセス<br>1＝16ラスタCGアクセス |
| ビット7 | 0＝アンダーラインなし<br>1＝アンダーラインあり |

**チェックポイント**　BASIC起動直後はビット1＝1である。これによりグラフィックを高解像度モニタで200ラインとして扱える(文字の方は400ライン)。turboでは，グラフィックが400ラインモードのときは偶数段目がバンク0，奇数段目がバンク1の内容が表示されている。ビット1が1であるということは，そのとき表示されているバンク(ビット3で指定)の内容のパターンを下の段にも繰り返し表示させるものである。

おそらくこの説明では理解できないだろうから，次のプログラムをサンプルとして示す。

```
10  OPTION SCREEN 0：WIDTH 80,25,0,2
20  GRAPH 0,0,0：CLS 4：SYMBOL(100,30),"バンク0",5,5,1,0,PSET
30  GRAPH 2,2,0：CLS 4：SYMBOL(100,70),"バンク1",5,5,2,0,PSET
40  OUT &H1FD0,&B1100001：A$=INKEY$(1)
50  OUT &H1FD0,&B1100011：A$=INKEY$(1)
60  OUT &H1FD0,&B1101001：A$=INKEY$(1)
70  OUT &H1FD0,&B1101011：A$=INKEY$(1)
```

また，turboではOUTだけだが，ZではINも可能。

ここは結構複雑である。ビット1はサンプルプログラムで理解して欲しい。ビット2はテキストの25（or 20）↔12（or 10）行の指定に使う。CRTCの設定と一緒に操作しなければ表示が乱れる。ビット3，4は簡単だと思われるが念のために言うと，640×400のモードのときビット3はどうでもよい。ビット1が0なら400ラインを表示する。ビット5はPCGのアクセス方式を変えるということで，PCGにアクセスする際には設定が必要である。

ビット6は8×8と8×16の2種類あるCGのどちらを読み込むかの指定である。画面表示には関係しない。ビット7はアンダーラインの指定。ここが1であり，アトリビュート（漢字VRAMのビット5）が1なら，（CRTCの設定も許すなら）アンダーラインが表示される。ちなみにアンダーラインの色はグラフィック1（青）の色に設定されているものが表示される。BASICのマニュアルにもあるように，

KSEN　　1，色

PALET　1，色

は同じ意味を持つ。これはアンダーラインモードを使った後はパレットを再設定すべきであることを示す。また気分が乗れば，PALET 命令の代わりに KSEN 命令を使うのもおつであろう。

### 1FE0H　黒色制御
### (IN)/OUT：turbo

| データ内容 | コントロール | |
|---|---|---|
| ビット0 | 黒変換するテキストの色を指定(0～7) | |
| ビット1 | | |
| ビット2 | | |
| ビット3 | テキストの黒変換のON/OFF | (1=ON, 0=OFF) |
| ビット4 | グラフィックの黒(透明)を黒変換 | |
| ビット5 | グラフィックの青を黒変換 | |
| ビット6 | ブランキング期間(枠)を黒変換 | |
| ビット7 | 未使用 | |

**チェックポイント**　グラフィックを黒変換するためには，パレットが0になっていなければならない。また，turbo ではOUT だけだが，Zではを INも可能。

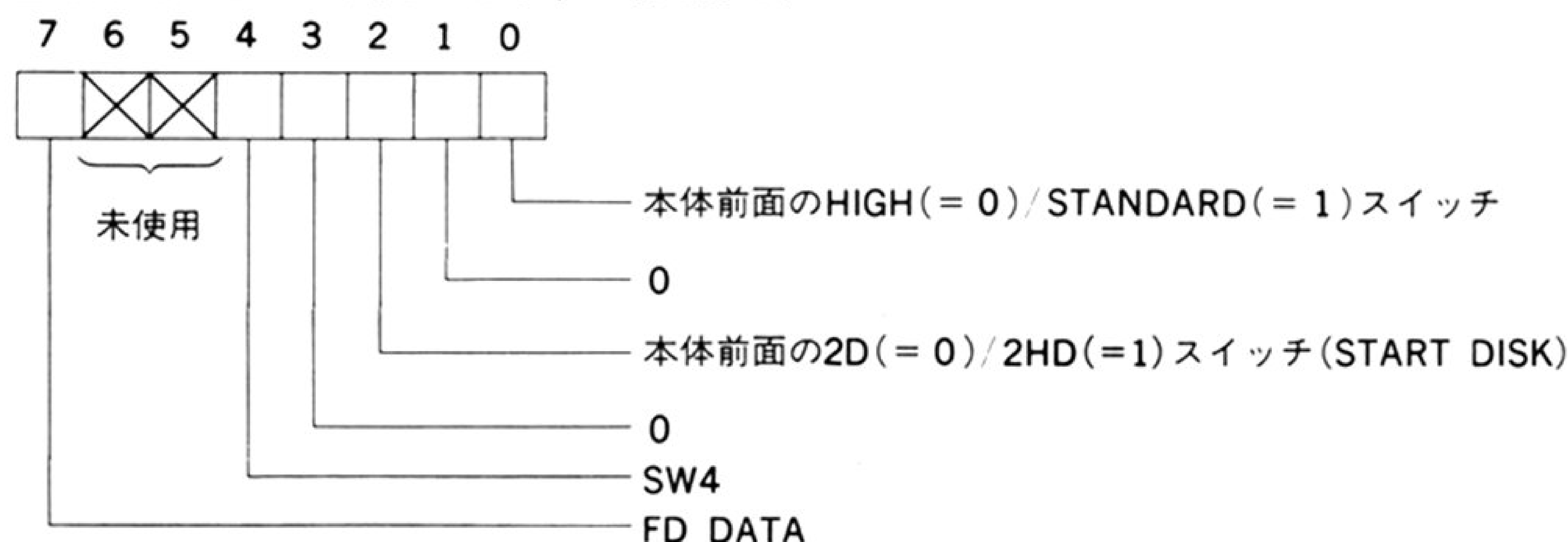
黒変換とは早い話がスーパーインポーズ時の黒抜きである。必要のない人には必要ないのだが，欲しい人には盆と正月がいっしょに来たようなものであろう。

### 1FF0H　スタートポート
### IN：turbo/Z

**turboおよび，Zの背面ディップスイッチSW5がOFF＝上側のとき**




**ZでSW5がON＝下側のとき(工場出荷時)**




**チェックポイント**　ビット1～3はBOOT時のディスクを指定する。SW4は意味を持たない。

| No. | SW3 | SW2 | SW1 | セレクト | 容　量 | 記録方式 | フォーマット |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 5(3)インチ | 320Kバイト | 2D　：両面倍密度 | |
| 1 | 0 | 0 | 1 | 5(3)インチ | 640Kバイト | 2DD　：両面倍密度 倍トラック | |
| 2 | 0 | 1 | 0 | 5インチ | 1Mバイト | 2HD　：両面高密度 | |
| 3 | 0 | 1 | 1 | 5インチ | 1Mバイト | 2HD　：両面高密度 | IBM |
| 4 | 1 | 0 | 0 | 8インチ | 1Mバイト | 2D256：両面倍密度 | |
| 5 | 1 | 0 | 1 | 8インチ | 1Mバイト | 2D256：両面倍密度 | IBM |
| 6 | 1 | 1 | 0 | 8インチ | 240Kバイト | 1S128：片面単密度 | IBM |
| 7 | 1 | 1 | 1 | ハードディスク | 10Mバイト | | |

これらのスイッチはただ単に付いているだけである。IPLなどのプログラムがここを見て，その設定にしたがった動作をするわけである。早い話が「フラグ」である。

背面ディップスイッチのSW4は無意味なので，ときにはOFF側に倒して気分を変えるのもよいだろう。

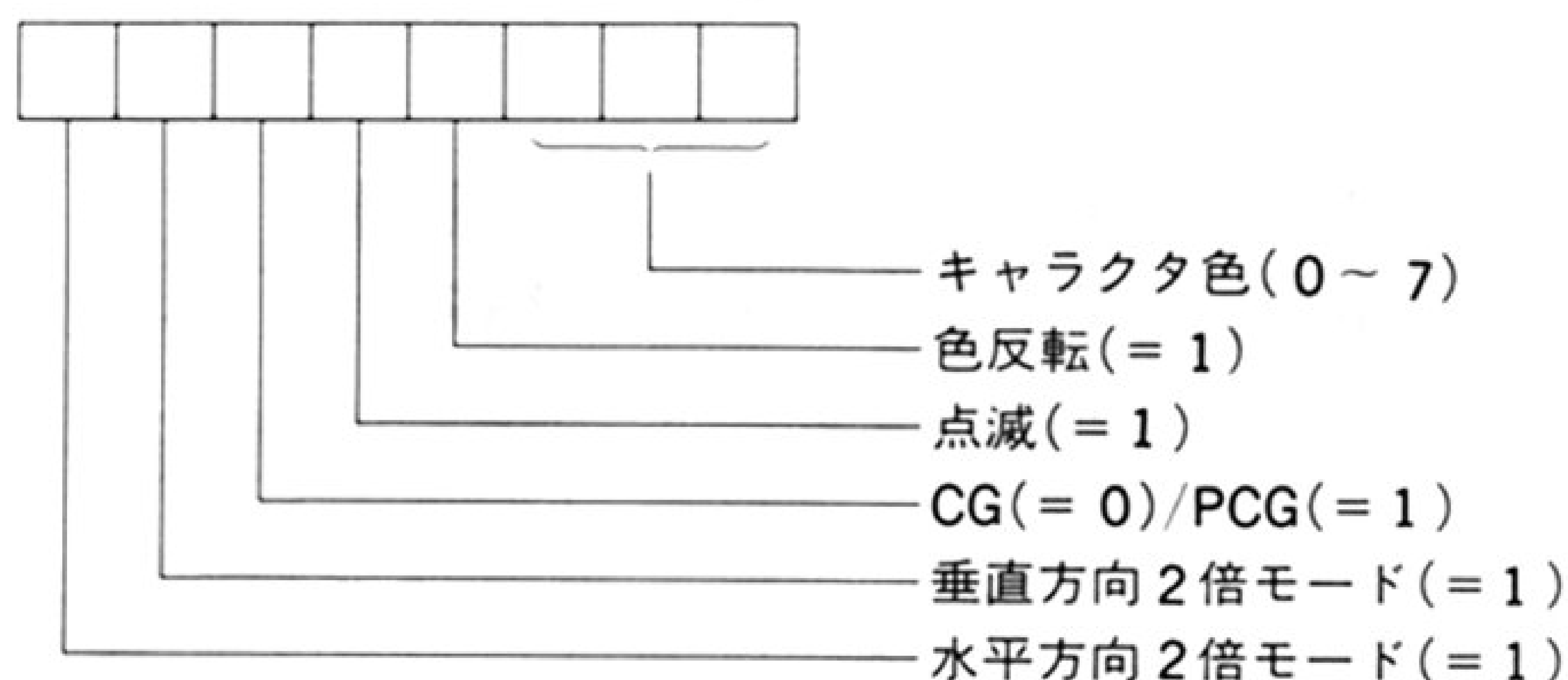
表中の2と3，4と5は同じではない。IBMフォーマットではサイド0，トラック0がFM(単密)記録方式になっている。なお，ビット7の「FD DATA」がその筋である。

ところで初期のころのturbo（市販版かどうかは定かではない）の中には，SW1〜SW4の順が逆になっているものもあるらしい。

ちなみにZの背面ディップスイッチSW6もディスクに関係している。詳しくはZのユーザーズマニュアルを参照。

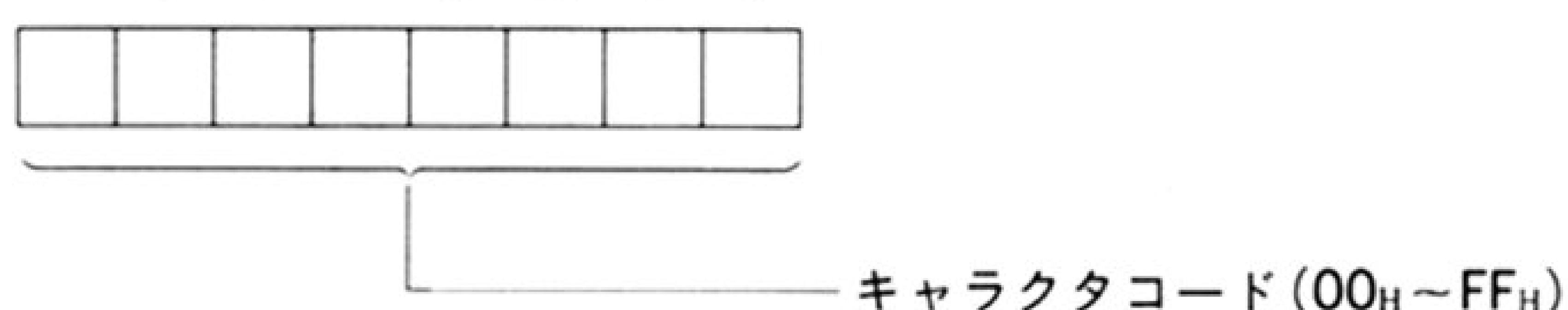
## $2000_H$〜$27FF_H$　テキストアトリビュート
## IN/OUT：X1

アトリビュート内のビットの意味




**チェックポイント**　当たり前のことだが，アトリビュートの内容が0なら色が黒。だから，なにも見えない。Zでは$1FB9_H$も参照。

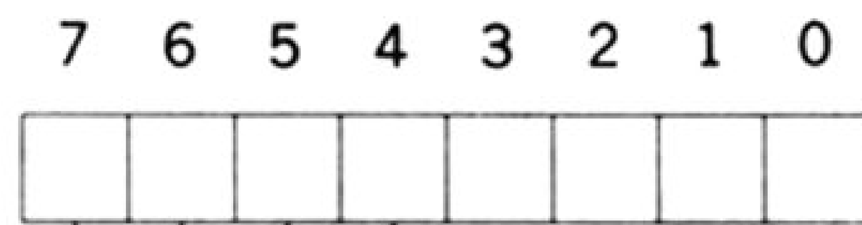
## $3000_H$〜$37FF_H$　テキストVRAM
## IN/OUT：X1/turbo




**チェックポイント**　turboで漢字を表示する場合は，漢字ROMアドレスの下位8ビット。

## 3800H～3FFFH　漢字VRAM（→第3章）

IN/OUT：turbo



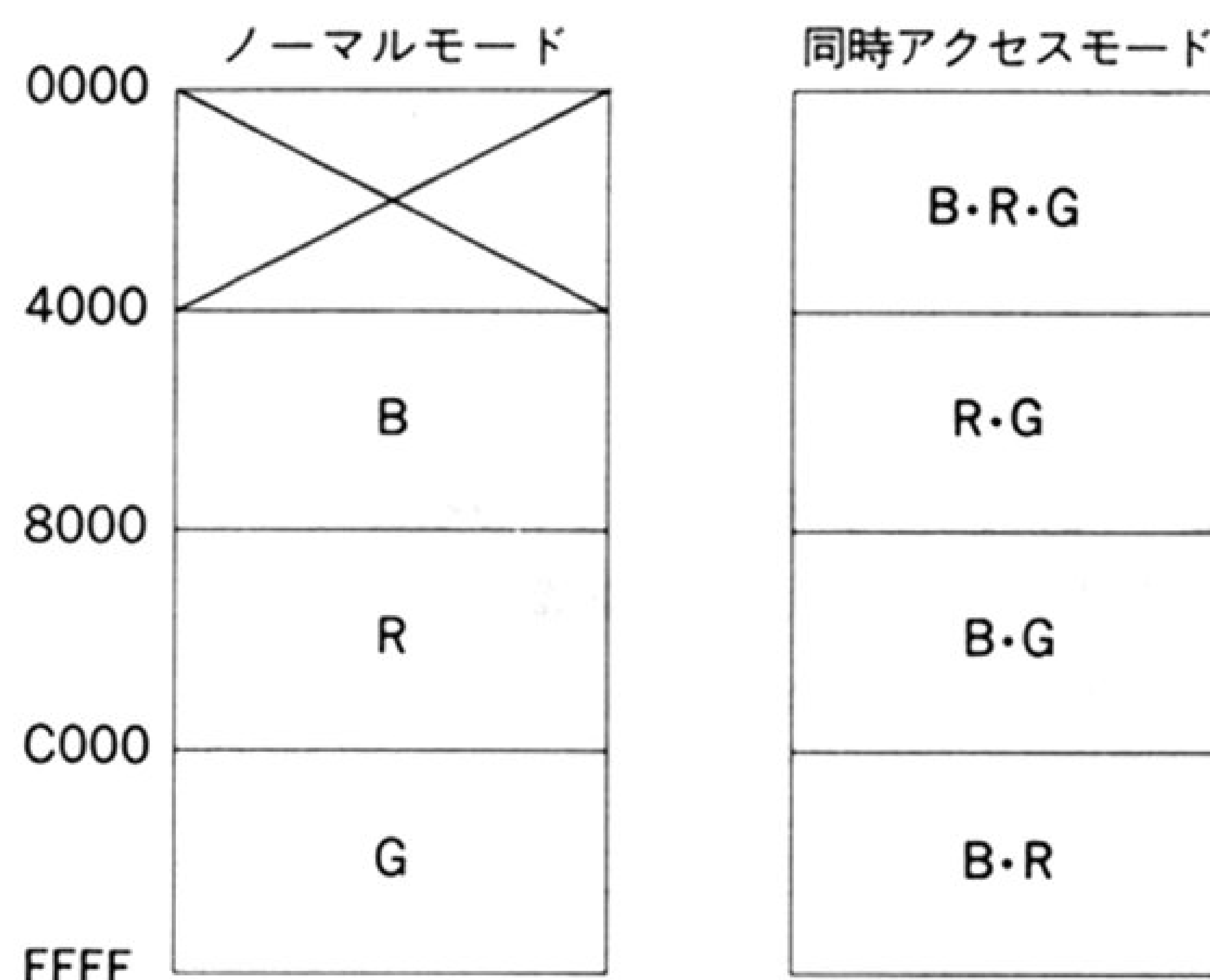

**チェックポイント**　下の表を見よ。

| テキストアトリビュート ビット5＝$\overline{\text{ROM}}$/RAM | 漢字VRAM ビット7＝$\overline{\text{CG}}$/漢字 | 漢字VRAM ビット4＝$\overline{1}$/2水準 | 表　示 |
|---|---|---|---|
| 0 | 0 | 無関係 | CG |
| 0 | 1 | 0 | 漢字(第1水準) |
| 0 | 1 | 1 | 漢字(第2水準) |
| 1 | 0 | 0 | PCG(ノーマル) |
| 1 | 0 | 1 | PCG外字① |
| 1 | 1 | 無関係 | PCG外字② |

PCG外字モードというのは，2個のPCGを使って8×16のキャラクタを構成するものである。4個のPCGで外字一つであるからturboでは計64個の外字（8色入り）を使えることになる。外字の出し方には2とおりあることが表からも分かるが，これはBASIC上で，全角（16×16）と半角（8×16）の二つを区別するために作られたものだそうである。

## 4000H～FFFFH　グラフィックRAM

IN/OUT：X1/turbo




**チェックポイント**　同時アクセスモードは1A02H，turboでのバンク切り換えは1FD0H。

X1のグラフィックマップは少々複雑であるが，次の式でBLUE上のアドレス（$4000_H$～$7FFF_H$）が計算できる。

X＝X座標，Y＝Y座標として640×200のとき，

&H4000＋(X¥8)＋((Y AND 7)＊2＾11)＋(Y¥8)＊80

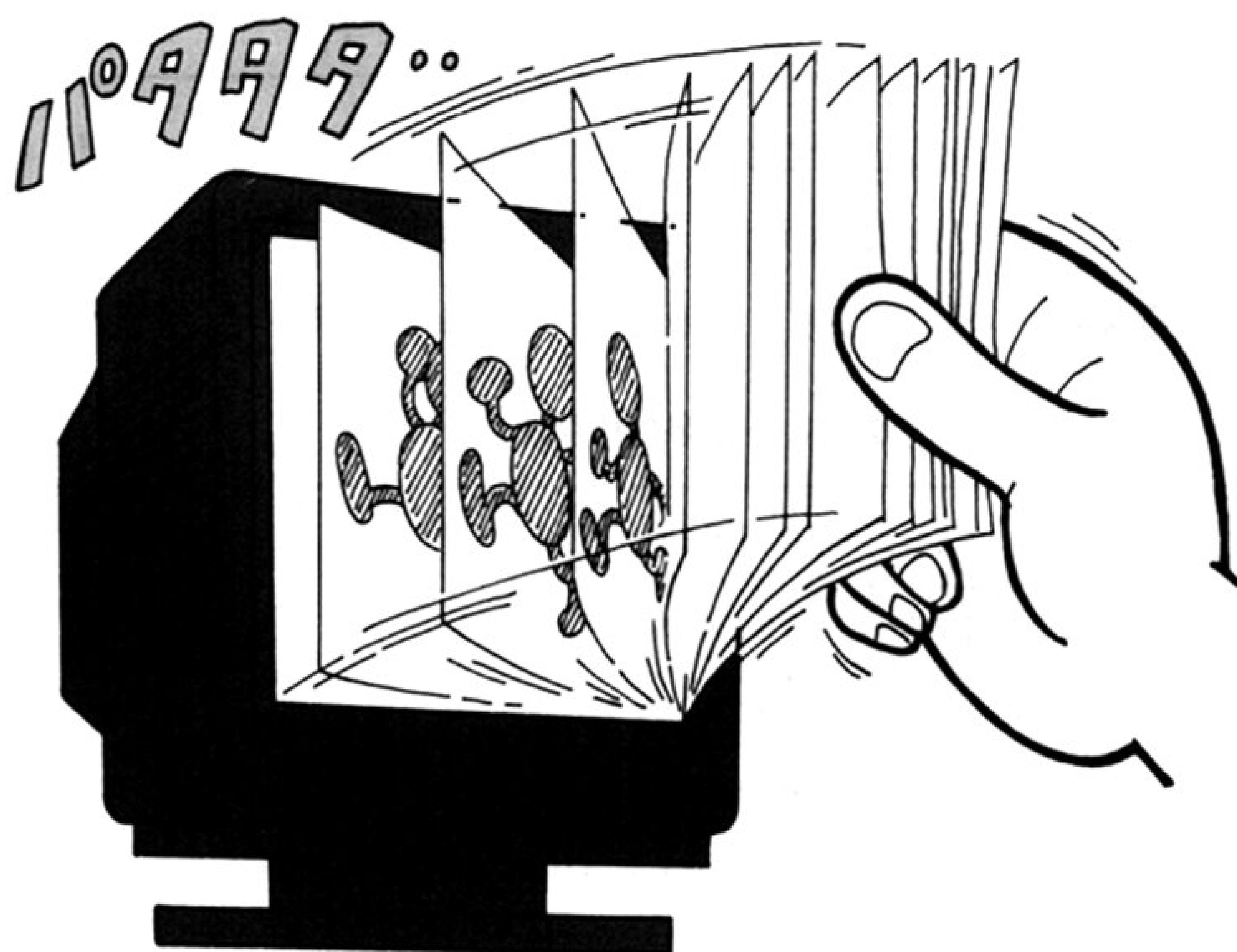
320×200の場合は最後の〝＊80〃を〝＊40〃に変えればよい。ここ以外はシフトで計算できるからマシン語で組むのは簡単である。

X1 turboの400ラインではバンク0とバンク1が1段おきになっていて，鋭く上位コンパチを保っている。400ラインの場合のアドレス計算は，Y AND 1をバンクNo.とし，Y＝Y¥2として，前述の値を計算すればよい。

また，Zでは4096色モード，64色モードが加わっているが，その色は，ページ0，1，2，3の順に薄い色になっている（ページとは，SCREEN，GRAPH命令で指定するやつである）。詳しくは第14章を参照。

# 第 1 章

# CRTC




CRTCでどすこいである

# 第1章
# CRTCでどすこいである・・・・・・・

この章ではCRTCをまな板に上げるのである。CRTCというのは，CRT Controller，すなわちCRT＝画面表示をコントロールするLSIのことなのである。つまりはキャラクタやらグラフィックやらを，CRTに送って表示する役割をしているのである。

まずは，CRTCがどのようなものかということを目で見るために，**リスト1-1**を実行していただきたい。突如現れる十文字である。しかも微妙なぼかしに，そこはかとない風情を感じるのは私だけではないだろう。このプログラムのネタは単純である。SCREENコマンドを使って画面に表示するページを，ページ0とページ1の間で高速に切り換えているために，二つの画面が重なって見えてしまうのである。ほら，何枚かの紙に少しずつ違う絵を描いて，パラパラとやると動いているように見えるというアニメーションの基本があるが，あれを二つの画面（ページ）でやっているのである。

**リスト1-1　ぼかし十文字**

```
100 WIDTH 40:INIT
110 GRAPH 0,0:CLS 4
120 PRINT STRING$(40,"-")
130 GRAPH 1,1:CLS 4
140 PRINT STRING$(40,"|")
150 GRAPH 0,0:GRAPH 1,1:GOTO 150
```

で，「BASICでこのようなことができるのであれば，機械語を使えばもっと美しいことになるのではないか」と思う人が少なからずいるであろう。しかし，**世の中はそう甘くない**ということを**リスト1-2**で噛みしめていただきたい。

**リスト1-2　ぼかし十文字機械語版**

```
100 WIDTH 40:INIT
110 GRAPH 0,0:CLS 4
120 PRINT STRING$(40,"-")
130 GRAPH 1,1:CLS 4
140 PRINT STRING$(40,"|")
150 '
160 CLEAR &HFE00:A=&HFE00
170 READ D$:IF D$="END" THEN 190
180 POKE A,VAL("&H"+D$):A=A+1:GOTO 170
190 CALL &HFE00
200 '
210 DATA 01,01,1A      :'        LD      BC,1A01H;8255
220 DATA D9            :'        EXX
230 DATA 01,00,18      :'        LD      BC,1800H
240 DATA 3E,0C         :'        LD      A,0CH
250 DATA ED,79         :'        OUT     (C),A    ;SET AR
260 DATA 0C            :'        INC     C        ;BC=1801H
266                    :';
270 DATA CD,1B,FE      :'LOOP:   CALL    EDGE
280 DATA AF            :'        XOR     A
```

```
285 DATA ED,79          :'        OUT     (C),A
290 DATA CD,1B,FE       :'        CALL    EDGE
300 DATA 3E,04          :'        LD      A,04H
305 DATA ED,79          :'        OUT     (C),A
310 DATA 18,F1          :'        JR      LOOP
320 'FE1B               :';
330 DATA D9             :'EDGE:   EXX
340 DATA ED,78          :'EDGE1:  IN      A,(C)
350 DATA F2,1C,FE       :'        JP      P,EDGE1
360 DATA ED,78          :'EDGE2:  IN      A,(C)
370 DATA FA,21,FE       :'        JP      M,EDGE2
380 DATA D9             :'        EXX
390 DATA C9             :'        RET
400 DATA END
```

リスト 1-2 の機械語ルーチンでは，垂直帰線期間のエッジ（立ち下がり）を検出し，ページの切り換えを行なっているのである（第 0 章の 8255 ②のポート B を参照)。もちろんそんなものを検出せず，機械語まかせに「たたたたた」とページ切り換えをしてもいいわけであるが，そうするともっと美しくないことになる（プログラムを変更できる人はやってみることをお勧めする)。それはともかくリスト 1-2 では機械語ルーチンの中に入った後は無限ループになっているので，止めるときはリセットスイッチ（NMI の方）を押していただきたい。それで結果であるが，BASIC でやったのと似たようなものである。すなわちこの「忍法二重画面」はあくまで「こんなこともできる」というだけなのである。

具体的な解説を行なう。表示画面の切り換えは，CRTC のレジスタの書き換えで行なうのである。すなわち表示開始アドレス（**CRTC にとっての** VRAM 上のアドレス）を変えてやるのである。しかし，いきなりそんなことを言われても理解不能であろう。それが普通なのである。よって，まずは CRTC の基本からやってみたいと思うのである。

# HD46505-SPなのだ

**表 1-1** を見ていただきたい。これは X1 に使われている CRTC，HD46505-SP のレジスタ機能表である（→参考文献 3)。この CRTC は 18 個（R0〜R17）までのレジスタを持ち，これらのレジスタに値を設定してやる（書き込んでやる）ことでいろいろな画面表示を実現するのである。表の右端には，普通の 40/80 文字モードにおける標準的な設定値を示しておいたので，**図 1-1** と併せて見ていただきたい。ただし，これは低解像度(200 ライン）の場合だから，X1 でならよいのだが，turbo BASIC（CZ-8FB02）で高解像度モード（400 ライン）を使っている場合ではちょっと都合が悪いので，その場合はすみやかに CZ-8FB01（もしくは CZ-8CB01）に立ち上げ直していただきたい。

それでは，できるだけ詳しく CRTC の内部レジスタの解説を行なうのである。なお，カッコの中の数字は「設定できる値の範囲」を示している。

**AR（0〜17)**

まずは隠れレジスタとして AR（アドレスレジスタ）がある。これは，内部レジスタ（R0〜R17）のうち，どれにアクセスするかを指定するためのものである（第 10 章の PSG へのアクセスと似ている)。一度設定すれば再設定しない限り AR は同じ値を保持している。

表 1-1　CRTC (HD46505-SP) 内部レジスタ機能表

| レジスタ番号 | 内　容 | 機　能 | 40桁 | 80桁 |
|---|---|---|---|---|
| R0 | 水平総文字数(－1) | 水平走査の周期を文字数に換算して指定する。 | 55 | 111 |
| R1 | 水平表示文字数 | 1行に表示する文字数を指定する。 | 40 | 80 |
| R2 | 水平同期位置(－1) | 水平同期信号を出力する位置(文字数)を指定する。 | 45 | 89 |
| R3 | 同期パルス幅 | 下位4ビットは水平同期信号のパルス幅を指定。<br>(文字数に換算)<br>上位4ビットは垂直同期信号のパルス幅を指定。<br>(水平走査線本数＝ラスタ本数に換算) | 52<br>($34_H$) | 56<br>($38_H$) |
| R4 | 垂直総文字数(－1) | 垂直走査の周期を文字数に換算して指定する。 | 31 | 31 |
| R5 | 総ラスタ調整 | 垂直走査の周期を微調整する。ラスタ数を指定する。 | 2 | 2 |
| R6 | 垂直表示文字数 | 表示する文字行数を指定する。 | 25 | 25 |
| R7 | 垂直同期位置(－1) | 垂直同期信号を出力する位置(文字数)を指定する。 | 28 | 28 |
| R8 | インタレース，スキュー | ラスタスキャンモード指定，DISPTMG信号，CUDISP信号のスキュー(遅れ)を指定する。 | 0 | 0 |
| R9 | 最大ラスタアドレス | 1行を構成するラスタ数(走査線)から1引いた値を指定する。 | 7 | 7 |
| (R10)<br>(R11) | カーソルスタートラスタ<br>カーソルエンドラスタ | カーソル形状，表示モードの指定。<br>X1では使われていない。 | －<br>－ | －<br>－ |
| R12<br>R13 | スタートアドレス上位<br>スタートアドレス下位 | 座標(0,0)に対応するテキストVRAMの値を指定する。<br>ただしVRAMは$0000_H$～$07FF_H$番地に対応する。 | 0<br>0 | 0<br>0 |
| (R14)<br>(R15) | カーソルアドレス上位<br>カーソルアドレス下位 | カーソルの表示アドレスを指定する。<br>X1では使われていない。 | －<br>－ | －<br>－ |
| (R16)<br>(R17) | ライトペン上位<br>ライトペン下位 | ライトペン検出アドレスを記憶する。<br>X1では使われていない。 | －<br>－ | －<br>－ |

$1800_H$番地に 0 ～17 を OUT すれば AR に値を設定できる。その後，$1801_H$に OUT すれば R0～R17 に値を設定できるわけだ。

**R0（0～255）**

**R1（0～255）**

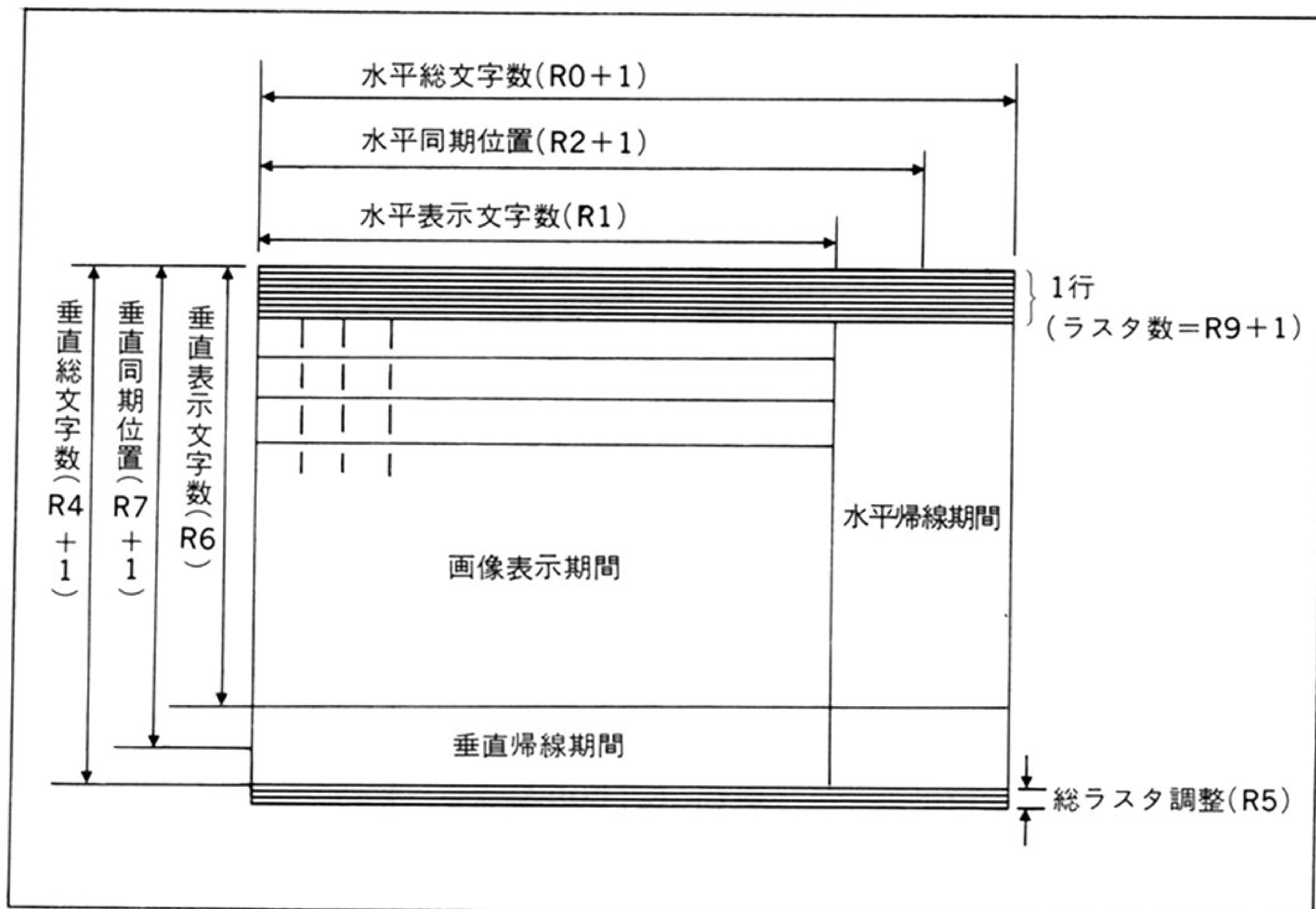
**R2（0～255）**

R0（水平総文字数）は水平走査周期を文字数に換算したものから－1したものである。総文字数というのは「実際に表示されている1行の文字数」＋「水平帰線時間を文字数に換算した値」のことである。R1 は表示文字数。これが「実際に表示される1行の文字数」である。R2 は水平同期位置から－1したもの（これも文字数に換算する)。

**R3（0～255）**

この1バイトは上位4ビットと下位4ビットの二つに分かれて意味を持つ。下位4ビットは水平同期信号のパルス幅を文字数に換算したもの。X1 では 40 桁で「4」, 80 桁で「8」を使っている。上位4ビットは垂直同期パルス幅をラスタ数に換算したものである。同様

図 1-1 CRTC 内部レジスタの意味




にX1では低解像度モニタ（標準解像度モニタ）に対しては「3」，turboでの高解像度モニタに対しては「8」を指定する。ところでいきなり「ラスタ」という言葉が出てきたが，これは「走査線」のことと思って差し支えない。ほら，200ラインのディスプレイをよーく見ると200本分の横縞が見えるであろう。アレである。

**R4（0～127）**

**R6（0～127）**

**R7（0～127）**

R4はR0，R6はR1，R7はR2と対応している。すべて文字数に換算して設定する。

**R5（0～31）**

これは垂直方向の帰線周期の微調整である。R4は文字数だが，R5はラスタ本数，すなわち走査線の本数で指定する。このレジスタはスーパーインポーズ時のスムーススクロール（BASICのスクロール命令）と関係している。第0章の8255②：$1A02_H$の説明を参照のこと。

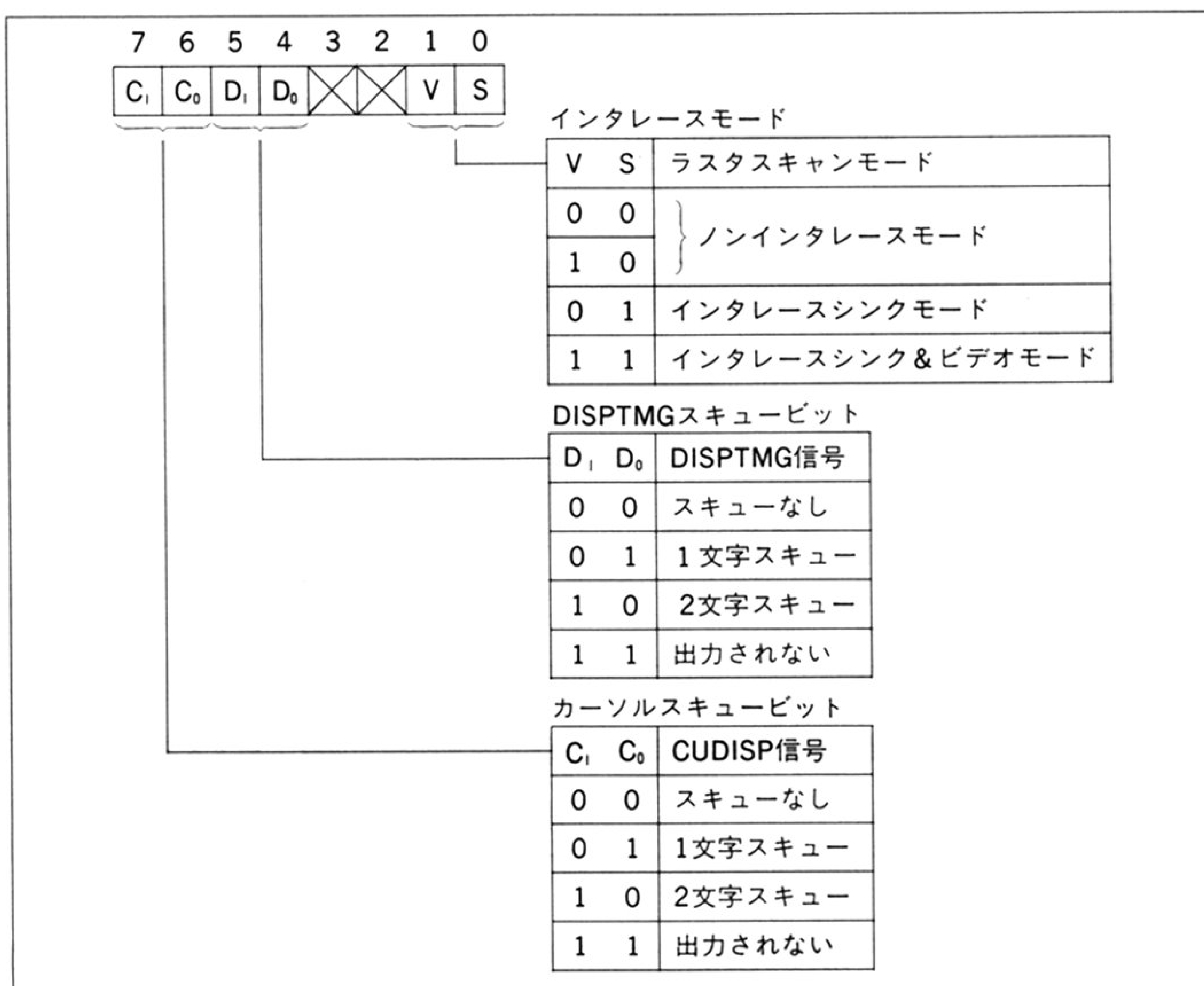
**R8**

**表 1-2** から分かるように，ビットごとに別々の意味を持つ。X1を普通に使う場合には常に0でよいが，それ以外の値を設定することも可能である。

**インタレースモード（ビット1，2）**

普通はノンインタレースモードである。インタレースシンクモードにすると画面が微妙に震えて，目を悪くするにはうってつけである。インタレースシンク&ビデオモードにすると，画面は縦にぐしゃっと潰れる（本当は潰れるだけではない）。そして，R5を調節しな

表 1-2　R8の機能




いと画面が上下にくるくる流れるであろう（垂直同期がずれるのだ）。

**DISPTMG 信号（ビット 5，4）**

普通はスキューなしのモードが設定されている。スキューを指定した場合は左端の 1 文字，もしくは 2 文字が CRT の右端に行ってしまう。その他の文字は移動しない。「11」で「出力されない」を指定すると，画面には何も現れなくなる。

**CUDISP 信号（ビット 7，6）**

X1 ではこの信号はオープン（無接続）になっているので，ここをいじっても何も起こらない。同じ CRTC を使っていた PASOPIA7 ではこの機能を使ってカーソルを出していた。

**R9（0～31）**

最大ラスタアドレスというのは，「1 文字を何ラスタで表示するか」なのだ。R9 にはその数－1 を設定する。X1 では文字は 8 ラスタで表示されるから 7 を指定する。turbo では 7 か 15，そしてアンダーラインモードのときは，アンダーラインの分も入れるので，9 か 19 となる。

**R10（0～127），R11（0～31）**

X1 では一切使っていない。カーソルの形状，モードの指定に使われる。

**R12（0～63）**

**R13（0～255）**

二つのレジスタを合わせた14ビットで表示開始アドレス（左上端に表示するキャラクタのテキストVRAMのアドレス）を指定する。X1では2048バイトのRAM（キャラクタ表示用）しか持っていない。これは2進数では&B00000000000～&B11111111111(11桁)であるからX1ではR12は下位3ビットだけに意味がある。最初に例を挙げたリスト1-1の画面切り換えは，R12，R13をいじっている。

**R14(0～654)，R15 (0～255)**

カーソルを表示するアドレスを指定する。X1では使われていない。

**R16(0～64)，R17 (0～255)**

ライトペンの位置を読み取るためのレジスタである。X1では使われていない。

ここで全体的なCRTCの解説の補足をしておく。

1) HD46505-SPの仕様では，R0～R11が書き込み可能，R12～R15が書き込み/読み込みの両方が可能，R16，R17が読み込みだけ可能となっている。ところがX1では読み込みは一切できない。

2) CRTCの説明では「文字」で機能を説明した。グラフィックは，「同じ位置に表示される文字」と同様な扱いを受ける。つまり，スタートアドレスを変えたならば，グラフィックも同様に表示が移動する。たとえば，テキストの$3000_H$番地と，G-RAMの$4000_H$，$4800_H$，$5000_H$，$5800_H$，$6000_H$，$6800_H$，$7000_H$，$7800_H$の8バイトは，CRTCをいかに設定しようとも重なり合って表示される。

以上である。ここらで気分転換に**リスト1-3**を打ち込んで遊んでいただきたい。このプログラムはR12，R13をいじってグラフィックを上下左右に8ドット単位でスクロールさせるものである。VRAMには触らずに，CRT上での表示位置（表示開始アドレス）を動かすことによってスクロールを実現している。念のために注意しておくが，これは「見て面白い」のであって，あまり実用にはならない。また，時々ちらつきが出るが，それは垂直帰線期間の間(つまり画面に何も表示されていない間)を考慮せず，R12，R13を設定しているからである。その気のある人はリスト1-2を参考にして正しく機械語で組んでみていただきたい。

**リスト1-3　ハードウェアスクロールもどき**

```
100 INIT:WIDTH 80:CLS 4:WIDTH 40
110 GRAPH 0,0
120 FOR J=1 TO 7:CIRCLE (280,160),J*5,J:NEXT
130 Y= 0:FOR X= 0 TO 30          :GOSUB "START":NEXT
140 X=30:FOR Y= 0 TO 15          :GOSUB "START":NEXT
150 Y=15:FOR X=30 TO  0 STEP -1:GOSUB "START":NEXT
160 X= 0:FOR Y=15 TO  0 STEP -1:GOSUB "START":NEXT
170 GOTO 130
180 '
190 LABEL "START"
200 A=X+Y*40:A1=A ¥ 256:A2=A MOD 256
210 'GRAPH 0,0
220 OUT &H1800,12:OUT &H1801,A1
230 OUT &H1800,13:OUT &H1801,A2
240 RETURN
```

# さらにCRTCを究める

**リスト 1-4, 1-5** は画面モードのサンプルである。

**リスト 1-4　X1 用画面サンプル**

```
100 INIT:CIRCLE(200,100),90
110 RESTORE "4025L"
120 FOR Z=1TO 8:GOSUB"MAIN":BEEP:WHILE(INKEY$(0)=""):WEND
130 NEXT:END
140 '
150 LABEL"FOO":WIDTH80:RESTORE"8025L":GOSUB"MAIN":BEEP
160 INPUT A$:ON ERROR GOTO 190:RESTORE A$:ON ERROR GOTO 0
170 GOSUB"MAIN":END
180 'ERROR TRAP
190 PRINT "ILLIGAL LABEL":BEEP:END
200 '
210 LABEL "BAR":WIDTH 80:RESTORE "8025L"
220 GOSUB"MAIN":END
230 '
240 LABEL"MAIN"
250 GOSUB"SETCRTC":GOSUB"80/40"
260 RETURN
270 '
280 LABEL"SETCRTC"
290 FORI=0TO13
300 READ D$:D=VAL("&H"+D$)
310 OUT&H1800,I:'SET CRTC REG. No.
320 OUT&H1801,D:'SET DATA
330 NEXT:RETURN
340 '
350 LABEL"80/40"
360 READ D$:D=VAL("&H"+D$)
370 OUT&H1A03,D
380 RETURN
390 '
400 LABEL"4025L":DATA 37,28,2D,34,1F,02,19,1C,00,07,00,00,00,00,0D:'40,25
410 LABEL"4020L":DATA 37,28,2D,34,18,08,14,16,00,09,00,00,00,00,0D:'40,20
420 LABEL"4012L":DATA 37,28,2D,34,0F,02,0C,0E,00,0F,00,00,00,00,0D:'40,12
430 LABEL"4010L":DATA 37,28,2D,34,0B,12,0A,0B,00,13,00,00,00,00,0D:'40,10
440 '
450 LABEL"8025L":DATA 6F,50,59,38,1F,02,19,1C,00,07,00,00,00,00,0C:'80,25
460 LABEL"8020L":DATA 6F,50,59,38,18,08,14,16,00,09,00,00,00,00,0C:'80,20
470 LABEL"8012L":DATA 6F,50,59,38,0F,02,0C,0E,00,0F,00,00,00,00,0C:'80,12
480 LABEL"8010L":DATA 6F,50,59,38,0B,12,0A,0B,00,13,00,00,00,00,0C:'80,10
490 '
500 LABEL"Q8006L":DATA 6F,50,59,38,07,02,06,07,00,1F,00,00,00,00,0C:'80,06
510 LABEL"Q8005L":DATA 6F,50,59,38,06,02,05,05,00,27,00,00,00,00,0C:'80,05
520 LABEL"Q8005D":DATA 6F,50,59,38,06,04,05,05,00,2F,00,00,00,00,0C:'80,05
530 LABEL"Q8003L":DATA 6F,50,59,38,03,02,03,03,00,3F,00,00,00,00,0C:'80,03
```

**リスト 1-5　X1 turbo 用画面サンプル**

```
100 INIT:CIRCLE(200,100),90
110 RESTORE "4025L"
120 FOR Z=1TO18:GOSUB"MAIN":BEEP:WHILE(INKEY$(0)=""):WEND
130 NEXT:END
140 '
150 LABEL"FOO":WIDTH80:RESTORE"8025H":GOSUB"MAIN":BEEP
160 INPUT A$:ON ERROR GOTO 190:RESTORE A$:ON ERROR GOTO 0
170 GOSUB"MAIN":END
180 'ERROR TRAP
190 PRINT "ILLIGAL LABEL":BEEP:END
200 '
210 LABEL "BAR":WIDTH 80:RESTORE "8025H"
220 GOSUB"MAIN":END
230 '
240 LABEL"MAIN"
250 GOSUB"SETCRTC":GOSUB"80/40":GOSUB"1FD*"
260 RETURN
270 '
280 LABEL"SETCRTC"
290 FORI=0TO13
300 READ D$:D=VAL("&H"+D$)
```

```
310 OUT&H1800,I:'SET CRTC REG. No.
320 OUT&H1801,D:'SET DATA
330 NEXT:RETURN
340 '
350 LABEL"80/40"
360 READ D$:D=VAL("&H"+D$)
370 OUT&H1A03,D
380 RETURN
390 '
400 LABEL"1FD*"
410 READ D$:D=VAL("&H"+D$)
420 OUT &H1FD0,D
430 RETURN
440 '
450 LABEL"4025L":DATA 37,28,2D,34,1F,02,19,1C,00,07,00,00,00,00,0D,00:'40,25
460 LABEL"4020L":DATA 37,28,2D,34,18,08,14,16,00,09,00,00,00,00,0D,80:'40,20
470 LABEL"4012L":DATA 37,28,2D,34,0F,02,0C,0E,00,0F,00,00,00,00,0D,04:'40,12
480 LABEL"4010L":DATA 37,28,2D,34,0B,12,0A,0B,00,13,00,00,00,00,0D,84:'40,10
490 '
500 LABEL"8025L":DATA 6F,50,59,38,1F,02,19,1C,00,07,00,00,00,00,0C,00:'80,25
510 LABEL"8020L":DATA 6F,50,59,38,18,08,14,16,00,09,00,00,00,00,0C,80:'80,20
520 LABEL"8012L":DATA 6F,50,59,38,0F,02,0C,0E,00,0F,00,00,00,00,0C,04:'80,12
530 LABEL"8010L":DATA 6F,50,59,38,0B,12,0A,0B,00,13,00,00,00,00,0C,84:'80,10
540 '
550 LABEL"4025H":DATA 35,28,2D,84,1B,00,19,1A,00,0F,00,00,00,00,0D,03:'40,25
560 LABEL"4025M":DATA 35,28,2D,84,1B,00,19,1A,00,0F,00,00,00,00,0D,01:'40,25
570 LABEL"4020H":DATA 35,28,2D,84,15,08,14,15,00,13,00,00,00,00,0D,81:'40,20
580 LABEL"4012H":DATA 35,28,2D,84,0D,00,0C,0D,00,1F,00,00,00,00,0D,07:'40,12
590 LABEL"4010M":DATA 35,28,2D,84,0D,00,0C,0D,00,1F,00,00,00,00,0D,05:'40,10
600 '
610 LABEL"8025H":DATA 6B,50,59,88,1B,00,19,1A,00,0F,00,00,00,00,0C,03:'80,06
620 LABEL"8025M":DATA 6B,50,59,88,1B,00,19,1A,00,0F,00,00,00,00,0C,01:'80,06
630 LABEL"8020H":DATA 6B,50,59,88,15,08,14,15,00,13,00,00,00,00,0C,81:'80,05
640 LABEL"8012H":DATA 6B,50,59,88,0D,00,0C,0D,00,1F,00,00,00,00,0C,07:'80,05
650 LABEL"8012M":DATA 6B,50,59,88,0D,00,0C,0D,00,1F,00,00,00,00,0C,05:'80,03
```

まずリスト 1-4 から解説しよう。これは X1 用であるが，当然のことに CZ-8FB01 を使って 200 ラインモードの画面であれば turbo でも使える。RUN させたなら，何かキーを押すたびに 8 とおりの画面モードが出現する。終わったなら，

　”BAR ⏎

と打ち込むと正常な画面になる。ただし，打ち込み間違いがあると表示がめちゃくちゃになるので，RUN させる前にセーブしておくべきである。

このプログラムの中心は 400〜480 行のデータ文である。サブルーチン〝MAIN〟は 14 個のデータを CRTC の内部レジスタ R0〜R13 までに設定する。最後の $0D_H$ もしくは $0C_H$ は 8255 に OUT されるデータである。これはビットセット/リセットモードを使っている。I/O の $1A02_H$ 番地のビット 6 を操作しているのだ。第 0 章の I/O マップを参照していただきたい。

X1 における正常な 40 桁，80 桁の画面はそれぞれ〝4025L〟と〝8025L〟に対応している。だから，それ以外のモードのときは実に楽しくシュールである。

これ以外の使い方としては，「”FOO ⏎」と打ち込むと，画面に「？」が出て入力モードになるはずである。このときラベル名を入れるとそのラベルの直後にあるデータを使って画面モードを設定することができる（たとえば？Q8006L ⏎）。このときは 500 行以降のデータも使えることになる。シュールさを味わって欲しい。ただし R5 の値が微妙であるから，場合によっては垂直同期などがずれて，表示が流れる可能性がある。いろいろ試してみるのも面白いかもしれない。

次に turbo 用のリスト 1-5 である。こちらの方は，1 行中のデータが一つ増えている。

これは 1FD0$_H$（画面管理ポート）への出力データである。問題になるのはビット 0～2 と 7 である。**表 1-3** を見て納得していただきたい。

以上が基礎編である。

**表 1-3　画面モードと画面管理ポート（1FD0$_H$）**

| | 低解像モード | | | | 高解像モード | | | | |
|---|---|---|---|---|---|---|---|---|---|
| テキスト | 25行 | 12行 | 20行 | 10行 | 25行 | 25行 | 12行 | 12行 | 20行 |
| グラフィック | 200ライン | 192ライン | なし | なし | 200ライン | 400ライン | 192ライン | 384ライン | なし |
| ビット0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| ビット1 | 0 | 0 | – | – | 1 | 0 | 1 | 0 | – |
| ビット2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| ビット7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

# もっと表示するのである

X1 で表示できる文字数は，最大で 80×25 の 2000 文字であると一般に信じられている。しかし，その認識は甘いのである。次のプログラム，**リスト 1-6** は 81×25，82×25 の画面モードを作ってしまうのだ。それぞれの場合について，よーく文字数を数えてみていただきたい。最後の 1 行は途中までしか表示していないが，これは BASIC 上でやっているので，勝手にスクロールさせないためなのだ。アドレスを計算して POKE@や OUT 文を使えばきっちり表示する（その場合はアトリビュートの設定も忘れずに）。

**リスト 1-6　WIDTH 81**

```
100 KEY 2,"WI.80"+CHR$(13)
110 L=81:' OR L=82
120 INIT:WIDTH 80:CLS 4
130 OUT &H1800,1:OUT &H1801,L
140 FOR I=1 TO 24:PRINT STRING$(L,CHR$(64+I));:NEXT
150 PRINT STRING$(80-(L-80)*25-1,CHR$(64+25));
160 GOTO 160
```

[F2] には〝WI. 80〟を定義してあるから，まともな画面に戻すときに使って欲しい。ただし turbo の場合は，CZ-8FB01 を使うこと。それから，CRT の調節の具合によっては，右端が画面からはみ出してしまうことがあるかもしれない。そのようなときは素直に納得していただきたい。

実行してみると分かるが，L=82 で 1 行 82 文字にしたとき，右下に「！」が 2 個表示される。これはなぜかというと，82×25=&H802=&H800+2 だからなのである。すなわち，CRTCは2050個の文字を表示しようとして，ともかくも2048個目まではどーっと表示するのである。ところがどっこい，テキスト VRAM はそこで終わりなので，CRTC は仕方なくテキスト VRAM の先頭に戻って 2 文字分をもう一度表示し，その場をつくろうのである。そう，CRTC は**どんな挑戦でも受けている**のである。

# 全画面である

最後にビデオマニアのために全画面ワイプ用のプログラムを**リスト 1-7** に示しておく。これは turboZ のオーバースキャンみたいなものと考えて差し支えないであろう。しつこいようだがturbo BASICでは動かないので，CZ-8FB01を使っていただきたい。

リスト 1-7　**全画面ワイプ**

```
100 GOSUB"BAR":' INITIALIZE
110 GOSUB"PCGDEF":' PCG DEFINITION
120 WIDTH80:CLS4:INIT
130 RESTORE"ALLWP":GOSUB"MAIN":' SET CRTC
140 CGEN1:COLOR7:'WIPE COLOR = 7 (WHITE)
150 FOR I=0 TO 8:CLS,I:PAUSE1:NEXT
160 FORI=8TO0STEP-1:CLS,I:PAUSE1:NEXT
170 IF INKEY$(0)="" THEN 150
180 GOSUB"BAR":END:' CRT=NORMAL AND END
190 '
200 LABEL"BAR":INIT:CLS,32:WIDTH40:RESTORE"4025L"
210 GOSUB"MAIN":RETURN
220 '
230 LABEL"MAIN"
240 GOSUB"SETCRTC":GOSUB"80/40"
250 RETURN
260 '
270 LABEL"SETCRTC"
280 FOR I=0TO13
290   READ D$:D=VAL("&H"+D$)
300   OUT &H1800,I:'SET CRTC REG. No.
310   OUT &H1801,D:'SET DATA
320 NEXT:RETURN
330 '
340 LABEL"80/40"
350 READ D$:D=VAL("&H"+D$)
360 OUT &H1A03,D
370 RETURN
380 '
390 LABEL"ALLWP":DATA 37,2E,30,34,1F,02,1C,1E,00,07,00,00,00,00,0D:'46,28
400 LABEL"4025L":DATA 37,28,2D,34,1F,02,19,1C,00,07,00,00,00,00,0D:'40,25
410 '
420 LABEL"PCGDEF"
430 M=&HFF
440 FOR I=0 TO 8
450   P$=STRING$(I,&H0)+STRING$(8-I,M)
460   DEFCHR$(I)=P$+P$+P$
470   M=INT(M/2)
480 NEXT:RETURN
```

プログラムの最初のイニシャライズでCRTCを正常にしているのは，PCG定義をするためである。PCGを定義するためには，CRTCが正しく普通のモードになっている必要があるのだ。詳しくは第2章の PCG を参照していただきたい。さて，プログラム中に

CLS, I

という命令が出てくるが，これは CZ-8FB01 の隠れ命令で，I の値を ASCII コードとするキャラクタをスペースとみなして画面をクリアするのである。一度この命令が実行されると，そのキャラクタが論理上のスペースとみなされるので，普通の女の子に戻ってまともな CLS をしたいときは，

CLS, 32

を実行する必要がある。これは turbo BASIC では削られている機能である。

このプログラムを実行すると，やがて画面は 46 字×28 行モードになる。しかし BASIC は 80×25 のモードのつもりで動いている。これによって「CLS，I」を実行すると，46 文

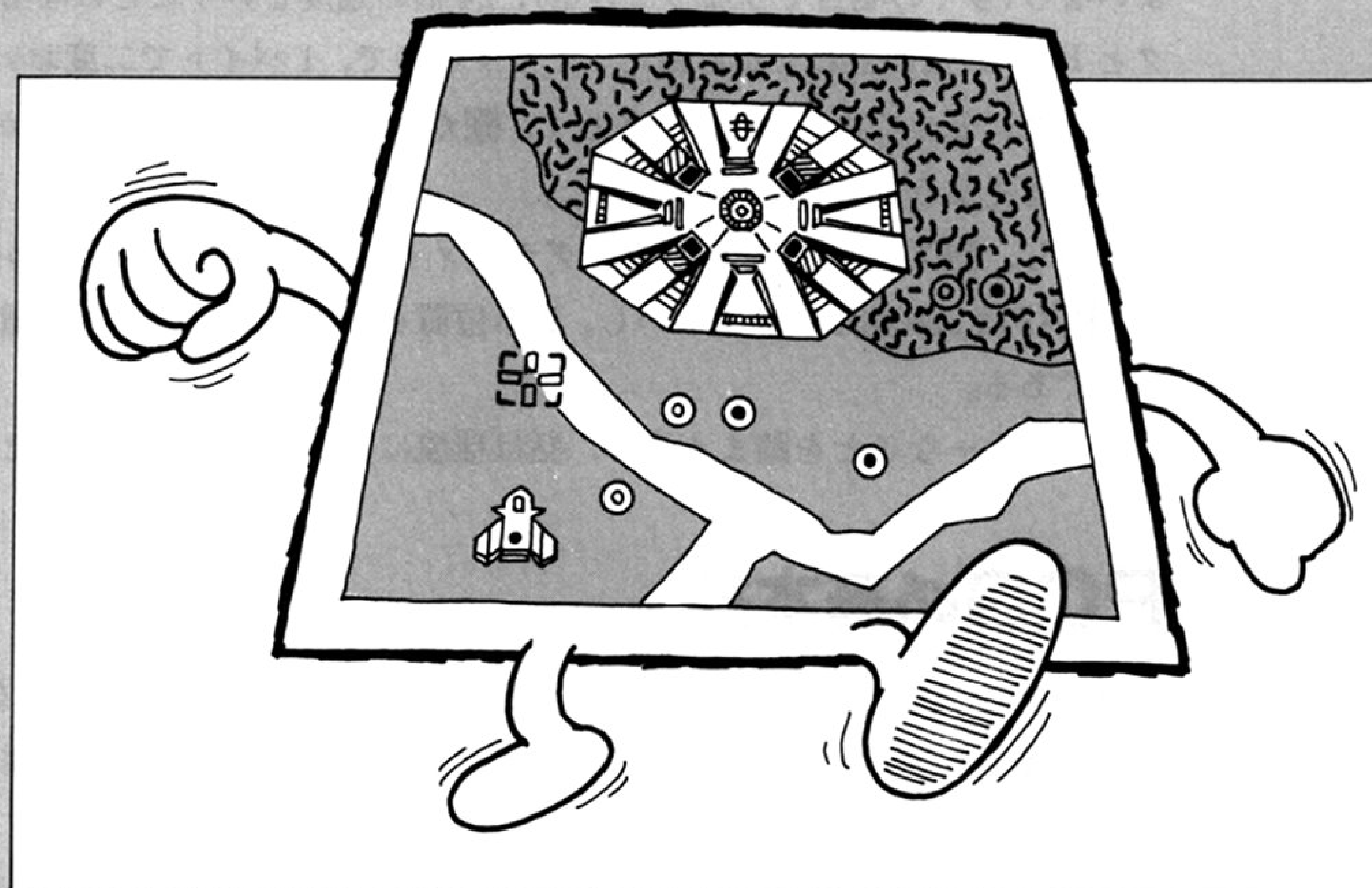
字×28 行をすべてキャラクタ＝CHR$(I)で埋めてくれる。プログラムはキーが押されるまでワイプを繰り返す。何かキーが押されたなら，画面を正常にして終了する。打ち込み間違いをすると画面がぐちゃぐちゃになるので注意すること。なお，このプログラムはスーパーインポーズ用である。スーパーインポーズでないときは画面の下の方に少し隙間ができているが，スーパーインポーズにすると，びょーんと伸びてくるのだ。これぞまさしく**東洋の神秘**と言わざるを得まい。

この章のサンプルプログラムでは充分に示せたかどうか分からないが，CRTC という石はかなり面白い。たとえば画面を 20×25 にして 4 面に分けて使うとかができる。しかし，実際に実用に値する裏技はあまりなく，私が考えついたのはリスト 1-7 の全画面ワイプぐらいなのである。これは，裏技は所詮裏技にすぎないということなのであろうか？

しかし X1 の CRTC はいじろうと思えばいじれるのである。そーなればとことんいじり回すのがその筋のその筋たるゆえんなのである。

# 第2章

# PCG




PCGは二度おいしいのである

# 第2章
# PCGは二度おいしいのである・・・

この章ではPCGをやるのである。念のために言っておくと，PCGとはProgrammable Character Generatorの意味である。ここで言うProgrammableとは「書き換えられるよーん」ということ。そしてCharacter Generatorとはそのままだと，「文字生成機」ということになる。文字生成機とは実に面妖であるが，つまりは「このASCIIコードの文字は，こーゆーパターンになっているのだ」ということを決めるものなのだ。Programmableでない場合は，そのパターン（FONT：フォント）はROMになっているわけである。そのような背景があるので，PCGは場合によっては，RAM CGなどと呼ばれることもあるのである。それの反対はROM CGである。

さて，周知のよーにX1やMZ-2500ではこのPCGがとてもパワフルなので，XEVIOUSやグラディウスなどがバシバシと動くのであった。PCGの利点は，(アトリビュートも入れるとすると) 2バイトのデータを転送するだけで，8ドット×8ドット×8色のパターンを移動させることができる，ということにある。グラフィックでやるなら，24バイトの転送であるから，単純に考えれば12倍の速度である。もし，アトリビュートを移動しなくてよいなら(多くの場合そうなっている)，24倍の速度ということになる。しかもグラフィックとPCGの重ね合わせが自由だったりするので，1バイトで二度おいしいのである。ゲーム製作者にとっては，これはまさしく，**棚から**PCG，もしくは，鴨がPCGを背負ってきた，と言えよう。

しかし，PCGといえども完全にグラフィックと置き換えられるわけではない。すなわち，PCGは256個しか使えないし，表示位置も8ドット単位でしか指定することができないのである。

以上のよーなことを踏まえつつ，話は佳境に入ってゆくのであった。

## PCGの基本技

BASIC上でPCGを使うには，まずはPCGを定義しなければならない。その後でモードを切り換えて表示するのである。すなわち，CZ-8FB01，8CB01では，

```
10 INIT：CLS
20 DEFCHR$(77)＝24バイトの文字列
30 CGEN 1
40 PRINT CHR$(77)
```

である。当たり前であった。

これを機械語的にやると，30～40行は大体，

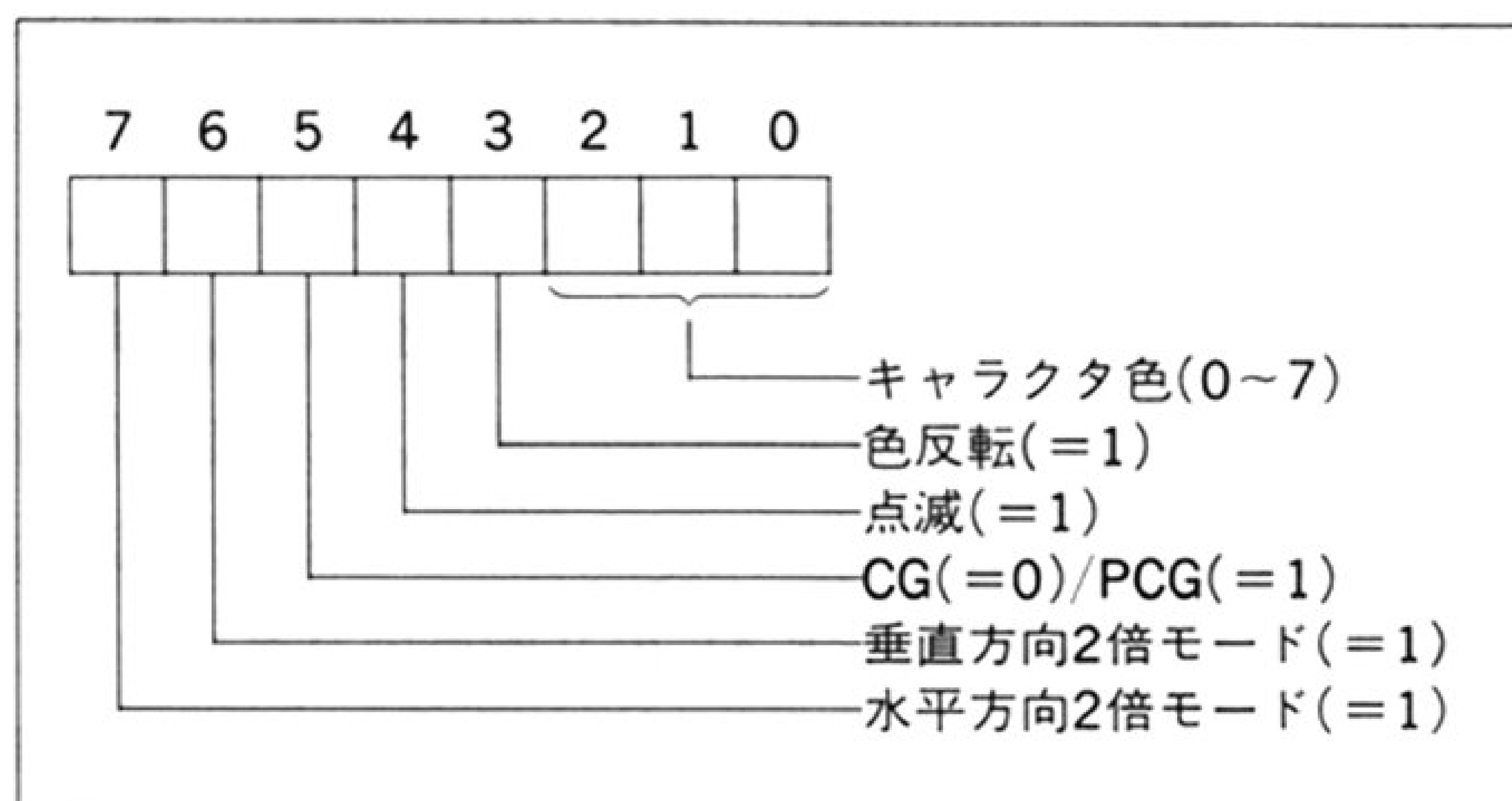
```
30 OUT &H2800，&H27
```

40　OUT　&H3000, 77

となる。やっていることは完全に同じではない点を注意していただきたい。

さて, 30 行の OUT 文はアトリビュートの設定である。アトリビュートの意味は第 0 章にも載せたが, ここでも**図 2-1** に示しておく。これにより, &H27 とは色が 7 であり, かつ PCG であるということが判明するであろう。40 行の 77 は単なる ASCII コードなわけだ。

**図 2-1　アトリビュート内のビットの意味**




アトリビュートを他のものにすれば, それぞれ横 2 倍, 縦 2 倍, 反転などなどになる。カラーを 7 以外にすると, PCG の色が変化する。たとえば**リスト 2-1** のように ASCII コード 77 の PCG に,

**青はA, 赤はB, 緑はC**

と定義しておいて, 色を 0 ～ 7 まで変化させて表示すると, それなりのパターンが表示される。試していただきたい。

**リスト 2-1　PCG とアトリビュートのカラー**

```
100 WIDTH 40
110 A$=""
120 FOR I=&H41 TO &H43
130   A$=A$+LEFT$(CGPAT$(I),8)
140 NEXT
150 DEFCHR$(77)=A$
160 FOR I=0 TO 7
170   OUT &H3000+I,77
180   OUT &H2000+I,&H20+I
190 NEXT
200 LOCATE 0,5
```

# PCGの定義

BASIC では, PCG の定義には DEFCHR$文を使うわけである。turbo では問題はないのだが, X1 では DEFCHR$は異常に遅い命令として津々浦々に知れ渡っている。以下に, なにゆえにそのよーに遅いのかを示しつつ, 超高速 PCG 定義などへと到達する予定である。

まずは手っ取り早く, 機械語で PCG を定義するプログラムを**リスト 2-2** に示す。**リスト**

**2-3** は例によって，BASIC からの使用方法である。リスト 2-2，2-3 は極めて基本技であるから，あくまでも学習用ということを心得ておいていただきたい。

リスト 2-2 を説明するに先立って，PCG 定義の原理を説明しておこう。

第 1 章でも示したように，X1 の画面表示は CRTC（CRT コントローラ）が取り仕切っているのである。ここでちょっと脇道にそれて，CRTC がどーやって文字を表示しているのかを，てきとーに考えてみるのである。CRTC はまず画面の左上から，横に走査線 1 本分（ラスタという）を表示するのである。CRTC は書き始めるに先立って，VRAM を見るのである（ここらへんは，極めていい加減な表現だから注意）。まずは左上の LOCATE 0，0 の位置だから，I/O 空間の $3000_H$番地にある VRAM の ASCII コードを見るわけだ。ここでは「A」つまり CHR$(&H41) が VRAM に書いてあるとしよう。すると CRTC は**「おっ！ $41_H$だな。それでは，キャラクタジェネレータの $41_H$に対応するところからパターンを持ってきて表示しましょう」**とつぶやくのである。ただし，「A」という文字は 8 × 8 ドットだから「上から何段目」ということも指定しなければならない。CRTC はそこんとこもちゃんと心得ていて，「ASCII コード $41_H$の 0 段目のパターン」というように指定してくるのである。この 8 ドットを表示した後は「VRAM の $3001_H$番地に書かれている ASCII コードの文字の 0 段目のパターン」，……と続くのである。そして，右上の端まで行ったら，同じ ASCII コードに対して段数を増加して 2 本目を左端から表示するのである。

注目して欲しいのは，これらの途中で，「CRTC が CG のパターン（データ）にアクセスしている」ということである。つまり，

**CRTC はパターンが格納されている ROM や RAM にアクセスしている**

のである。そこで PCG が，ある VRAM にアクセスしているとき，その VRAM に定義の対象になる ASCII コードを用意しておくと，CRTC は律儀にその ASCII コードに対応する CG ROM（または RAM）にアクセスする。そのタイミングをねらって，**別の回路からデータを送ってやり，パターンを書き込んでしまう**のである。これが X1 における PCG 定義の仕組みである。

以上が原理なのだが，よーく考えてみると落とし穴があるのだ。どこにあるかというと，PCG が CRTC にアクセスされてデータが読み出されている最中に，横からシャシャリ出てきて，その PCG に読み書き動作をするなど許されないということなのだ。普通のメモリに対して「読むことと書くことを同時に行なう」のは許されないのである（現在はデュアルポート RAM といって，それができる RAM もあるが）。

では，本当のところはどーやっているのかというと，

**CRTC が「表示されない VRAM」にアクセスしているときだけ，他の回路からのデータ R/W を許している**

のである。

そのココロは，VRAM は $3000_H$〜$37FF_H$＝2048 バイトあるのに，画面表示に使われてる VRAM は，普通は，最大で 80×25＝2000 バイトで，48 バイトが宙に浮いているということなのである。CRTC がその VRAM に空しいアクセスをしているスキをねらって，デー

タの読み書きを行なうのである。その「表示はされないが，CRTCからアクセスされるVRAM」というのは，

1） WIDTH　40：SCREEN，0の場合
   I/Oの$33E8_H$～$33FF_H$までの24バイト
2） WIDTH　40：SCREEN，1の場合
   I/Oの$37E8_H$～$37FF_H$までの24バイト
3） WIDTH　80の場合
   I/Oの$37D0_H$～$37FF_H$までの48バイト

の三つの場合がある。実際はそれぞれに対応するアトリビュート(turboの場合は，さらに漢字VRAMも）セットしておく必要がある。

そこでやっとリスト2-2，2-3の説明に入るのであった。

**リスト2-2　機械語によるPCG定義**

```
                                        .Z80
                                        .PHASE  0E000H
                                ;
  E000    EB                            EX      DE,HL
  E001    46                            LD      B,(HL)   ;HI
  E002    23                            INC     HL
  E003    4E                            LD      C,(HL)   ;LOW
  E004    23                            INC     HL
  E005    ED 43 E078                    LD      (BCWORK),BC
  E009    56                            LD      D,(HL)   ;COUNT
  E00A    23                            INC     HL
  E00B    5E                            LD      E,(HL)   ;ASCII CODE
  E00C    23                            INC     HL
  E00D    CD E018                       CALL    SET0
                                ;
  E010    46                            LD      B,(HL)
  E011    23                            INC     HL       ;HL POINTS PCG DATA
  E012    0E 00                         LD      C,00H    ;BC=1500H,1600H or 1700H
  E014    CD E052                       CALL    SETPCG
  E017    C9                            RET
                                ;
  E018    E5                    SET0:   PUSH    HL       ;SAVE POINTER
  E019    D5                            PUSH    DE       ;SAVE COUNT and ASCII CO
DE
  E01A    01 1FD0                       LD      BC,1FD0H          ;FOR turbo
  E01D    AF                            XOR     A
  E01E    ED 79                         OUT     (C),A
                                ;
  E020    ED 4B E078                    LD      BC,(BCWORK)
  E024    21 3800                       LD      HL,3800H          ;KANJI VRAM(turb
o)
  E027    3E 00                         LD      A,00H
  E029    CD E048                       CALL    SET1
                                ;
  E02C    ED 4B E078                    LD      BC,(BCWORK)
  E030    21 2800                       LD      HL,2800H          ;ATTRIBUTE
  E033    D1                            POP     DE
  E034    D5                            PUSH    DE
  E035    3E 20                         LD      A,20H             ;PCG,COLOR=0
  E037    CD E048                       CALL    SET1
                                ;
  E03A    ED 4B E078                    LD      BC,(BCWORK)
  E03E    21 3000                       LD      HL,3000H          ;VRAM
  E041    D1                            POP     DE
  E042    7B                            LD      A,E               ;ASCII CODE
  E043    CD E048                       CALL    SET1
                                ;
  E046    E1                            POP     HL
  E047    C9                            RET
                                ;
  E048    09                    SET1:   ADD     HL,BC
  E049    44                            LD      B,H
  E04A    4D                            LD      C,L      ;BC=HL+BC
  E04B    ED 79                 SET2:   OUT     (C),A
```

```
E04D     03                      INC     BC        ;INC ADDRESS
E04E     15                      DEC     D         ;DEC COUNTER
E04F     20 FA                   JR      NZ,SET2
E051     C9                      RET
                        ;
E052     1E 08          SETPCG:  LD      E,08H     ;COUNTER
E054     D9                      EXX
                        ;
E055     F3                      DI
E056     01 1A01                 LD      BC,1A01H
E059     ED 78          VDISP0:  IN      A,(C)
E05B     F2 E059                 JP      P,VDISP0
E05E     ED 78          VDISP1:  IN      A,(C)
E060     FA E05E                 JP      M,VDISP1
                        ;
E063     D9                      EXX
E064     7E             SETP:    LD      A,(HL)    ;7,GET DATA
E065     ED 79                   OUT     (C),A     ;12,SET 1 BYTE
E067     23                      INC     HL        ;6,INC POINTER
E068     03                      INC     BC        ;6,INC I/O ADDRESS
                        ;7+12+6+6=31
                        ;
E069     00                      NOP               ;4
E06A     23                      INC     HL        ;6
E06B     2B                      DEC     HL        ;6,DUMMY
                        ;4+6+6=16
                        ;
E06C     3E 0D                   LD      A,0DH     ;7
E06E     3D             DLY:     DEC     A         ;4
E06F     C2 E06E                 JP      NZ,DLY    ;10
E072     1D                      DEC     E         ;4
E073     C2 E064                 JP      NZ,SETP   ;10
                        ;7+(4+10)*13+4+10=203
                        ;
                        ;31+16+203=250
E076     FB                      EI
E077     C9                      RET
                        ;
E078                    BCWORK:  DS      2
                                 END
```

リスト 2-2 で最初にやっているのは，パラメータの受け取りである。例によって DE レジスタを利用している。まずは DE レジスタの内容を HL レジスタに移して，BC レジスタに格納している。これは先程説明した「表示されない VRAM」の先頭アドレス(相対値) である。念のために書くが，なぜいちいち受け取っているかというと，そのアドレスは画面モードによって違ってくるからである。その次に D レジスタに格納している「COUNT」も画面モードによって変わるのは同じことである。

次に肝心の ASCII コードである。これは以上のパラメータを持って SET0 をコールしている。SET0 では ($1FD0_H$) ← $00_H$としているが，これは turbo 用である。turbo ではここが画面管理を行なっており，X1 と同じ動作をさせるためには，$00_H$を OUT しておく必要がある。ただしこれは念のためであって，X1 モードで起動しておけば，IPL が自動的に $00_H$をセットしておいてくれるから深く悩む必要はない。それは次の漢字 VRAM でも同じで，$3800_H$＋BC レジスタから，D レジスタ分 $00_H$をセットしている。もし turbo でなかった場合は，これは $3000_H$＋BC レジスタ番地からの VRAM にセットされることになるが，その場合は直後にまた VRAM へ ASCII コードをセットするので，ただ単にムダになるだけであって，画面が乱れるなどの悪影響はないので安心するよーに。それからアトリビュートも忘れてはいけない。こいつは $20_H$ をセットしているから，つまりはノーマルの PCG で，色は 0（つまり透明）である。ここは第 5 ビットが立っていれば何でもよいので，**気分によっては** $FF_H$などにしてもよいだろう。

SET0はこれぐらいで，次は御本尊のSETPCGである。まず$E010_H$で，あらかじめBレジスタにパラメータを渡している点に注意。この時点でBレジスタには，$15_H$，$16_H$，$17_H$のうちどれかが入り，それによって青，赤，緑のうちどれがセットされるかが決まるのだ。**図2-2**を見て納得していただきたい。

図2-2　CG，漢字ROM，PCGアクセス用I/Oポート

X1，X1 turboコンパチアクセスモード

| | | |
|---|---|---|
| $14**_H$ | CG ROMアクセス | IN |
| $15**_H$ | PCG BLUEアクセス | IN/OUT |
| $16**_H$ | PCG REDアクセス | IN/OUT |
| $17**_H$ | PCG GREENアクセス | IN/OUT |

X1 turbo高速アクセスモード

| | | |
|---|---|---|
| $14*0_H$～$14*F_H$ | CG，漢字ROMアクセス | IN |
| $15*0_H$～$15*F_H$ | PCG BLUEアクセス | IN/OUT |
| $16*0_H$～$16*F_H$ | PCG REDアクセス | IN/OUT |
| $17*0_H$～$17*F_H$ | PCG GREENアクセス | IN/OUT |

**チェックポイント**
turboの場合，I/Oポート$1FD*_H$の第5ビットが1ならば高速アクセスモード，0ならばコンパチアクセスモード。

さて，よく見ると分かるよーに，実はこの方式では，一度に1色分しかセットできないのである。すなわち，正しくPCGをセットするには，このルーチンを3回呼び出さなければならないのだ。この点を心得ておいて欲しいわけである。

では$E052_H$からのSETPCGである。まずEレジスタに$08_H$をロードしている。これはとりもなおさず，キャラクタのデータが8段飾りであることに由来する。つまり$08_H$というのはカウンタなのである。その後$E055_H$～$E062_H$で何やら行なっているが，これは「垂直帰線期間信号の立ち下がり」（I/Oポート$1A01_H$番地の第7ビット）の検出なのである。「そんなもんを検出して，一体どーすんだ」と思うであろうが，これには「今からCRTCは例の表示されないVRAMへアクセスを開始しま～す」という意味があるのだ。よってこれがGOサインなのである。ところで，見慣れない条件JPがあるが，これは各自で自習していただきたい。

いよいよ核心中の核心が，$E064_H$からである。まず，(BC)へ1バイトOUTしている。これでPCGセットが1段分できたことになる。$E067_H$までは実にありふれたプログラムである。そして，出ました，ディレイルーチンである。プログラムの右端に並んでいるのは，ステート数（クロック数：X1は4MHzだから1クロックが$0.25\mu s=0.00000025$秒）である。ループ1回分が250クロックになるようにしてある。なぜこのよーになっているかというと，これはCRTCの都合によるものなのだ。すなわち「PCGのセットはCRTCが表示されないRAMにアクセスしているときを利用する」という大原則があるので，「CRTC様が次の段にアクセスなされるとき」をひたすら待たなければならないのである。なぜ250クロックかというと，これは映像信号の水平周期＝15.98 kHzからきている

のだ。すなわち，

$$\frac{1}{15.98(\mathrm{kHz})} \fallingdotseq 0.0000625\ (\text{秒})$$

$$= \frac{250\ (\text{クロック})}{4(\mathrm{MHz})}$$

なのだ。15.98 kHz というのがどこから出てきたかというと，資料に書いてあったからなのだ。私は深く追及する気はないので，さっさとリスト 2-3 の説明に入ってしまうのである。

**リスト 2-3　低速 PCG 定義**

```
100 CLEAR &HE000
110 MEM$(&HE000,16)=HEXCHR$("EB 46 23 4E 23 ED 43 78 E0 56 23 5E 23 CD 18 E0")
120 MEM$(&HE010,16)=HEXCHR$("46 23 0E 00 CD 52 E0 C9 E5 D5 01 D0 1F AF ED 79")
130 MEM$(&HE020,16)=HEXCHR$("ED 4B 78 E0 21 00 38 3E 00 CD 48 E0 ED 4B 78 E0")
140 MEM$(&HE030,16)=HEXCHR$("21 00 28 D1 D5 3E 20 CD 48 E0 ED 4B 78 E0 21 00")
150 MEM$(&HE040,16)=HEXCHR$("30 D1 7B CD 48 E0 E1 C9 09 44 4D ED 79 03 15 20")
160 MEM$(&HE050,16)=HEXCHR$("FA C9 1E 08 D9 F3 01 01 1A ED 78 F2 59 E0 ED 78")
170 MEM$(&HE060,16)=HEXCHR$("FA 5E E0 D9 7E ED 79 23 03 00 23 2B 3E 0D 3D C2")
180 MEM$(&HE070,16)=HEXCHR$("6E E0 1D C2 64 E0 FB C9 00 00 00 00 00 00 00 00")
190 DEFUSR0=&HE000
200 '
210 CGEN 1:LOCATE 0,0
220 FORI=0TO255:PRINT #0,CHR$(I);:NEXT:PRINT
230 CGEN 0
240 '
250 TIME=0
260 FORI=0TO255
270   A$=USR0(CHR$(&H7,&HD0,48,I,&H15)+STRING$(8,&H3))
280   A$=USR0(CHR$(&H7,&HD0,48,I,&H16)+STRING$(8,&HC))
290   A$=USR0(CHR$(&H7,&HD0,48,I,&H17)+STRING$(8,&H30))
300 NEXT
310 PRINT TIME
320 TIME=0
330 FORI=0TO255:DEFCHR$(I)=STRING$(24,0):NEXT
340 PRINT TIME
350 '
360 ' A$=USR0(CHR$(&H7,&HD0,48,?,&H??)+STRING$(8,&H??)):WIDTH 80
370 ' A$=USR0(CHR$(&H7,&HE8,24,?,&H??)+STRING$(8,&H??)):WIDTH 40:SCREEN ?,0
380 ' A$=USR0(CHR$(&H7,&HE8,24,?,&H??)+STRING$(8,&H??)):WIDTH 40:SCREEN ?,1
```

まず 210〜230 行ですべての PCG を表示しておく。次に 260〜300 行ですべての PCG を 3 色縦縞にしてしまう。CHR$（△）の中のパラメータは，WIDTH 80 のときのものである(他のスクリーンモードのときは 360 行からの注釈文を見ておくれ)。そして，すぐ後では DEFCHR$文で同じようなことをやっている。結局は速度比較なのである。で，その速度はどーかというと，これがなんと同じで，両方とも 256 個の PCG を定義するのに 13 秒前後かかっているのである。考えてみれば，CRT の垂直同期周波数は 60Hz 前後である。つまり，1 秒間に 60 回 PCG をセットするチャンスが来るわけだ。256 個セットするには，1 個当たり青赤緑の 3 回のチャンスが必要だから，256×3＝768 回のチャンスが必要である。そいでもって，768÷60＝12.8（秒）かかってしまうのである。

そこで出てくるのが，3 倍速 PCG 定義プログラムであった。

# PCGの高速定義

まずは，3 倍の速さにするのである。この原理は私が考えたものではないということを

最初に断っておく。

**リスト 2-4, 2-5** を見ていただきたい。注意してみると分かるように，リスト 2-4 の前半はリスト 2-2 とほとんど同じである。違ってくるのは，E04E$_H$の SETPCG からである。まずは ORDER というサブルーチンを呼んでいる。ここで何をやっているかというと，青，赤，緑と 8 バイトずつのブロックになっているデータを，青の 0 段目，赤の 0 段目，緑の 0 段目，……という順序に並べ変えて，WORK（E0AC$_H$）から始まる 24 バイトに格納しているのである。E08D$_H$～E0A3$_H$あたりの 2 重ループの作り方が実に微妙なのである。特に注意すべきことは「LDI」を実行して BC レジスタが 0 になったかどうかの判断を，「JP　PO，～」（パリティ・オッド）で判定している点である。「LDI」はゼロフラグを変えないのである。

**リスト 2-4　3 倍速 PCG 定義の機械語部分**

```
                                        .Z80
                                        .PHASE   0E000H
                                ;
 0016                           BLUE    EQU      15H+1
 0017                           RED     EQU      16H+1
 0018                           GREEN   EQU      17H+1
                                ;
 E000    EB                             EX       DE,HL
 E001    46                             LD       B,(HL)   ;HI
 E002    23                             INC      HL
 E003    4E                             LD       C,(HL)   ;LOW
 E004    23                             INC      HL
 E005    ED 43 E0AA                     LD       (BCWORK),BC
 E009    56                             LD       D,(HL)   ;COUNT
 E00A    23                             INC      HL
 E00B    5E                             LD       E,(HL)   ;ASCII CODE
 E00C    23                             INC      HL
 E00D    CD E014                        CALL     SET0
                                ;
                                ;HL POINTS PCG DATA
 E010    CD E04E                        CALL     SETPCG
 E013    C9                             RET
                                ;
 E014    E5                     SET0:   PUSH     HL        ;SAVE POINTER
 E015    D5                             PUSH     DE        ;SAVE COUNT & ASCII CODE
 E016    01 1FD0                        LD       BC,1FD0H         ;FOR turbo
 E019    AF                             XOR      A
 E01A    ED 79                          OUT      (C),A
                                ;
 E01C    ED 4B E0AA                     LD       BC,(BCWORK)
 E020    21 3800                        LD       HL,3800H         ;KANJI VRAM:turb
o
 E023    3E 00                          LD       A,00H
 E025    CD E044                        CALL     SET1
                                ;
 E028    ED 4B E0AA                     LD       BC,(BCWORK)
 E02C    21 2800                        LD       HL,2800H         ;ATTRIBUTE
 E02F    D1                             POP      DE
 E030    D5                             PUSH     DE
 E031    3E 20                          LD       A,20H            ;PCG,COLOR=0
 E033    CD E044                        CALL     SET1
                                ;
 E036    ED 4B E0AA                     LD       BC,(BCWORK)
 E03A    21 3000                        LD       HL,3000H         ;VRAM
 E03D    D1                             POP      DE
 E03E    7B                             LD       A,E              ;ASCII CODE
 E03F    CD E044                        CALL     SET1
                                ;
 E042    E1                             POP      HL
 E043    C9                             RET
                                ;
 E044    09                     SET1:   ADD      HL,BC
 E045    44                             LD       B,H
 E046    4D                             LD       C,L      ;BC=HL+BC
 E047    ED 79                  SET2:   OUT      (C),A
 E049    03                             INC      BC       ;INC ADDRESS
 E04A    15                             DEC      D        ;DEC COUNTER
```

```
E04B     20 FA                   JR      NZ,SET2
E04D     C9                      RET
                        ;
                        ;Acc=ASCII
E04E     CD E086        SETPCG:  CALL    ORDER    ;CHANGE ORDER
                        ;
E051     06 16                   LD      B,BLUE
E053     0E 00                   LD      C,00H
E055     16 17                   LD      D,RED    ;RED
E057     1E 18                   LD      E,GREEN  ;GREEN
E059     3E 08                   LD      A,08H    ;COUNTER
E05B     08                      EX      AF,AF'
E05C     D9                      EXX
                        ;
E05D     F3                      DI
E05E     01 1A01                 LD      BC,1A01H
E061     ED 78          VDSP0:   IN      A,(C)
E063     F2 E061                 JP      P,VDSP0
E066     ED 78          VDSP1:   IN      A,(C)
E068     FA E066                 JP      M,VDSP1
                        ;
E06B     D9                      EXX
E06C     08                      EX      AF,AF'   ;A'=COUNTER
                        ;
E06D     ED A3          SETP:    OUTI             ;16,OUT BLUE
E06F     42                      LD      B,D      ;4
E070     ED A3                   OUTI             ;16,OUT RED
E072     43                      LD      B,E      ;4
E073     ED A3                   OUTI             ;16,OUT GREEN
                        ;16+4+16+4+16=56
                        ;
E075     06 16                   LD      B,BLUE   ;7
                        ;7=7
                        ;
E077     08                      EX      AF,AF'   ;4
E078     3E 0B                   LD      A,0BH    ;7
E07A     3D             DLY:     DEC     A        ;4
E07B     C2 E07A                 JP      NZ,DLY   ;10
E07E     08                      EX      AF,AF'   ;4
                        ;4+7+(4+10)*11+4=169
                        ;
E07F     0C                      INC     C        ;4
E080     3D                      DEC     A        ;4
E081     C2 E06D                 JP      NZ,SETP  ;10
                        ;4+4+10=18
                        ;56+7+169+18=250
E084     FB                      EI
E085     C9                      RET
                        ;
                        ;CHANGE ORDER
                        ;B0,B1,...,B7,R0,R1,...,R7,G0,G1,...,G7
                        ;      ---->B0,R0,G0,B1,R1,G1,...,B7,R7,G7
                        ;
E086     D9             ORDER:   EXX
E087     06 08                   LD      B,8      ;COUNTER
E089     D9                      EXX
E08A     11 E0AC                 LD      DE,WORK  ;BUFFER
E08D     E5             ORDER0:  PUSH    HL
E08E     01 0003                 LD      BC,3
E091     ED A0          ORDER1:  LDI
                        ;(DE)<-(HL),INC DE,INC HL,DEC BC
E093     E2 E09E                 JP      PO,ORDER2        ;3 BYTES MOVED?
E096     79                      LD      A,C      ;SAVE C
E097     0E 07                   LD      C,7
E099     09                      ADD     HL,BC    ;HL=HL+7
E09A     4F                      LD      C,A      ;BACK C
E09B     C3 E091                 JP      ORDER1
                        ;
E09E     E1             ORDER2:  POP     HL
E09F     23                      INC     HL
E0A0     D9                      EXX
E0A1     05                      DEC     B        ;DEC COUNTER
E0A2     D9                      EXX
E0A3     C2 E08D                 JP      NZ,ORDER0
                        ;
E0A6     21 E0AC                 LD      HL,WORK  ;NEW PCG DATA
E0A9     C9                      RET
                        ;
E0AA                    BCWORK:  DS      2
E0AC                    WORK:    DS      24
                                 END
```

リスト 2-5　3 倍速 PCG 定義

```
100 CLEAR &HE000
110 MEM$(&HE000,16)=HEXCHR$("EB 46 23 4E 23 ED 43 AA E0 56 23 5E 23 CD 14 E0")
120 MEM$(&HE010,16)=HEXCHR$("CD 4E E0 C9 E5 D5 01 D0 1F AF ED 79 ED 4B AA E0")
130 MEM$(&HE020,16)=HEXCHR$("21 00 38 3E 00 CD 44 E0 ED 4B AA E0 21 00 28 D1")
140 MEM$(&HE030,16)=HEXCHR$("D5 3E 20 CD 44 E0 ED 4B AA E0 21 00 30 D1 7B CD")
150 MEM$(&HE040,16)=HEXCHR$("44 E0 E1 C9 09 44 4D ED 79 03 15 20 FA C9 CD 86")
160 MEM$(&HE050,16)=HEXCHR$("E0 06 16 0E 00 16 17 1E 18 3E 08 08 D9 F3 01 01")
170 MEM$(&HE060,16)=HEXCHR$("1A ED 78 F2 61 E0 ED 78 FA 66 E0 D9 08 ED A3 42")
180 MEM$(&HE070,16)=HEXCHR$("ED A3 43 ED A3 06 16 08 3E 0B 3D C2 7A E0 08 0C")
190 MEM$(&HE080,16)=HEXCHR$("3D C2 6D E0 FB C9 D9 06 08 D9 11 AC E0 E5 01 03")
200 MEM$(&HE090,16)=HEXCHR$("00 ED A0 E2 9E E0 79 0E 07 09 4F C3 91 E0 E1 23")
210 MEM$(&HE0A0,16)=HEXCHR$("D9 05 D9 C2 8D E0 21 AC E0 C9 00 00 00 00 00 00")
220 DEFUSR0=&HE000
230 '
240 CGEN 1:LOCATE 0,0
250 FORI=0TO255:PRINT #0,CHR$(I);:NEXT:PRINT
260 CGEN 0
270 '
280 TIME=0
290 FORI=0TO255
300   A$=USR0(CHR$(&H7,&HD0,48,I)+STRING$(24,1))
310 NEXT
320 PRINT TIME
```

このよーにして並べ変えたデータを $E06D_H$ から OUTI を使って，走査線 1 本当たり，青，赤，緑の 3 バイトずつ PCG にセットしているわけである。ここでの注意点は，OUTI 命令のひねくれ方である。実に**ムッとすることに**，OUTI は，B レジスタを一つ減少させてから，OUT 動作を行なうのである。よって，$1500_H$ 番地に OUT したいならば，BC を $1600_H$ にしておかなければならないのである。むむむーん。あと特筆すべきことは，その 3 バイトを OUT する期間である。どうやら 60 クロックぐらいの間に済まさなければならないよーである。しかし，私はその根拠は計算できない。ま，56 クロックで収まっているのだからよしとしてしまうのである。

この 3 倍速 PCG 定義を使うと，256 個の PCG 定義が 4 秒強で済むことになる。しかしこの世にその筋として生を受けたからには，「もっとオイシイ手はないかかかっ！」と考えなければならん。そして思い付いたのが，24 倍速 PCG 定義である。一体どーやってやるのかというと，原理は単純で，

**PCG 定義で大事な「表示されない VRAM」の数（行数）を増やしてしまおう。**

という，実にその筋的な発想なのである。第 1 章で CRTC をやったが，CRTC の第 6 レジスタ（R6）は「垂直表示文字数」であった。普段はここは 25 なのだが，7 減らして 18 にしてしまうと，どーだ「表示されない VRAM」がどっと増えただろーが。7 行増えたのだから，3 倍速× 8 倍速で 24 倍速である。機械語部分が**リスト 2-6**，BASIC から使っているのが**リスト 2-7** である。リスト 2-6 でやっているのは，CRTC をいじっているほかは，リスト 2-4 の 3 倍速 PCG 定義と大差ない。そこでさっさとリスト 2-7 の説明に行ってしまうのである。

リスト 2-6　24 倍速 PCG 定義の機械語部分

```
                                    .Z80
                                    .PHASE  0E000H
                            ;
 0016                       BLUE    EQU     15H+1
 0017                       RED     EQU     16H+1
 0018                       GREEN   EQU     17H+1
                            ;
```

```
E000    01 1800                 LD      BC,1800H        ;SET CRTC
E003    3E 06                   LD      A,6
E005    ED 79                   OUT     (C),A
E007    03                      INC     BC
E008    3E 12                   LD      A,18    ;DISP JUST 18 LINES
E00A    ED 79                   OUT     (C),A
                        ;
E00C    01 1FD0         START:  LD      BC,1FD0H        ;FOR turbo
E00F    AF                      XOR     A
E010    ED 79                   OUT     (C),A
                        ;
E012    01 3DA0                 LD      BC,3800H+5A0H   ;KANJI VRAM
E015    21 0260                 LD      HL,260H ;COUNTER
E018    16 00                   LD      D,00H
E01A    CD E070                 CALL    SETIO
                        ;
E01D    01 25A0                 LD      BC,2000H+5A0H   ;ATTRIBUTE
E020    21 0260                 LD      HL,260H ;COUNTER
E023    16 20                   LD      D,20H
E025    CD E070                 CALL    SETIO
                        ;
E028    21 E200                 LD      HL,0E200H       ;DATA AREA
E02B    22 E0AF                 LD      (HLWORK),HL     ;SAVE ADDRESS
E02E    AF                      XOR     A       ;TOP OF ASCII CODE
E02F    F5              LOOP:   PUSH    AF
E030    CD E056                 CALL    SET8
E033    2A E0AF                 LD      HL,(HLWORK)     ;DATA ADDRESS
E036    CD E07A                 CALL    SETPCG
E039    2A E0AF                 LD      HL,(HLWORK)
E03C    01 00C0                 LD      BC,24*8
E03F    09                      ADD     HL,BC           ;NEW ADDRESS
E040    22 E0AF                 LD      (HLWORK),HL
E043    F1                      POP     AF
E044    C6 08                   ADD     A,8             ;CHECK COUNTER
E046    C2 E02F                 JP      NZ,LOOP
                        ;
E049    01 1800                 LD      BC,1800H
E04C    3E 06                   LD      A,6
E04E    ED 79                   OUT     (C),A
E050    03                      INC     BC
E051    3E 19                   LD      A,25    ;CRTC WO MODOSU
E053    ED 79                   OUT     (C),A
E055    C9                      RET
                        ;
E056    01 35A0         SET8:   LD      BC,3000H+5A0H
E059    57                      LD      D,A     ;ASCII CODE
E05A    1E 08                   LD      E,8     ;COUNTER
E05C    C5              SET80:  PUSH    BC
E05D    21 0030                 LD      HL,48
E060    CD E070                 CALL    SETIO
E063    C1                      POP     BC
E064    21 0050                 LD      HL,80
E067    09                      ADD     HL,BC
E068    44                      LD      B,H
E069    4D                      LD      C,L     ;BC=NEXT ADDRESS
E06A    14                      INC     D       ;INC ASCII CODE
E06B    1D                      DEC     E       ;DEC COUNTER
E06C    C2 E05C                 JP      NZ,SET80
                        ;
E06F    C9                      RET
                        ;
E070    ED 51           SETIO:  OUT     (C),D
E072    03                      INC     BC
E073    2B                      DEC     HL
E074    7C                      LD      A,H
E075    B5                      OR      L
E076    C2 E070                 JP      NZ,SETIO
E079    C9                      RET
                        ;
                        ;Acc=ASCII
E07A    06 16           SETPCG: LD      B,BLUE
E07C    0E 00                   LD      C,00H
E07E    16 17                   LD      D,RED   ;RED
E080    1E 18                   LD      E,GREEN ;GREEN
E082    3E 40                   LD      A,08H*8 ;COUNTER,8 CHARS!
E084    08                      EX      AF,AF'
E085    D9                      EXX
                        ;
E086    F3                      DI
E087    01 1A01                 LD      BC,1A01H
E08A    ED 78           VDSP0:  IN      A,(C)
```

```
E08C     F2 E08A               JP      P,VDSP0
E08F     ED 78         VDSP1:  IN      A,(C)
E091     FA E08F               JP      M,VDSP1
                       ;
E094     D9                    EXX
E095     08                    EX      AF,AF'   ;A'=COUNTER
                       ;
E096     ED A3         SETP:   OUTI             ;16,OUT BLUE
E098     42                    LD      B,D      ;4
E099     ED A3                 OUTI             ;16,OUT RED
E09B     43                    LD      B,E      ;4
E09C     ED A3                 OUTI             ;16,OUT GREEN
                       ;16+4+16+4+16=56
                       ;
E09E     06 16                 LD      B,BLUE   ;7
                       ;7=7
                       ;
E0A0     08                    EX      AF,AF'   ;4
E0A1     3E 0B                 LD      A,0BH    ;7
E0A3     3D            DLY:    DEC     A        ;4
E0A4     C2 E0A3               JP      NZ,DLY   ;10
E0A7     08                    EX      AF,AF'   ;4
                       ;4+7+(4+10)*11+4=169
                       ;
E0A8     0C                    INC     C        ;4
E0A9     3D                    DEC     A        ;4
E0AA     C2 E096               JP      NZ,SETP  ;10
                       ;4+4+10=18
                       ;56+7+169+18=250
E0AD     FB                    EI
E0AE     C9                    RET
                       ;
E0AF                   HLWORK: DS      2
                               END
```

リスト 2-7　24 倍速 PCG 定義

```
100 CLEAR &HE000
110 MEM$(&HE000,16)=HEXCHR$("01 00 18 3E 06 ED 79 03 3E 12 ED 79 01 D0 1F AF")
120 MEM$(&HE010,16)=HEXCHR$("ED 79 01 A0 3D 21 60 02 16 00 CD 70 E0 01 A0 25")
130 MEM$(&HE020,16)=HEXCHR$("21 60 02 16 20 CD 70 E0 21 00 E2 22 AF E0 AF F5")
140 MEM$(&HE030,16)=HEXCHR$("CD 56 E0 2A AF E0 CD 7A E0 2A AF E0 01 C0 00 09")
150 MEM$(&HE040,16)=HEXCHR$("22 AF E0 F1 C6 08 C2 2F E0 01 00 18 3E 06 ED 79")
160 MEM$(&HE050,16)=HEXCHR$("03 3E 19 ED 79 C9 01 A0 35 57 1E 08 C5 21 30 00")
170 MEM$(&HE060,16)=HEXCHR$("CD 70 E0 C1 21 50 00 09 44 4D 14 1D C2 5C E0 C9")
180 MEM$(&HE070,16)=HEXCHR$("ED 51 03 2B 7C B5 C2 70 E0 C9 06 16 0E 00 16 17")
190 MEM$(&HE080,16)=HEXCHR$("1E 18 3E 40 08 D9 F3 01 01 1A ED 78 F2 8A E0 ED")
200 MEM$(&HE090,16)=HEXCHR$("78 FA 8F E0 D9 08 ED A3 42 ED A3 43 ED A3 06 16")
210 MEM$(&HE0A0,16)=HEXCHR$("08 3E 0B 3D C2 A3 E0 08 0C 3D C2 96 E0 FB C9 00")
220 '
230 WIDTH 80
240 CGEN 1
250 FOR I=0 TO 255:PRINT #0 CHR$(I);:NEXT:PRINT
260 CGEN 0
270 FOR I=0 TO 255
280   MEM$(&HE200+I*24,24)=STRING$(24,I)
290 NEXT
300 TIME=0
310 FOR I=0 TO 10:CALL &HE000:NEXT
320 PRINT TIME
```

リスト 2-7 では最初に WIDTH を 80 にしている。そーなのである。リスト 2-6 は簡単にするために，WIDTH 80 でしか動かないよーになっているのだ。しかも PCG 用のデータは，あらかじめ青，赤，緑，青，赤，緑，……というように並べ換えられて，$E200_H$から 6144 ($1800_H$) バイトに格納されていることになっている。270〜290 行で，そのダミーのデータを作っているわけである。実際の PCG 定義は「CALL &HE000」一発で OK である。速すぎるので 10 回実行している。結果は，256 個の定義にかかる時間は約 0.6 秒と出た。さて，理論的には全画面を表示されない VRAM（？）にして 26×３＝78 倍速なんていうのができそうだが，ちょっと無理のようである。実は 16×３＝48 倍速までやってみた

のだが，ソフトウェアでタイミングをとる悲しさで，所々でPCGの定義に失敗してしまうのだ。それで安全のために24倍速でとどめておいたのである(その筋な某ユーザーは，もう一工夫して，うまくソフトウェアでタイミングをとり，48倍速定義を成功させたそうである。ももも)。

# CGの読み出し

話は変わって，BASICのCGPAT$関数に対応する部分である。**リスト2-8，2-9**にROM CGを読み出すプログラムを示す。リスト2-2〜7を参考にすれば，PCGから読み出すことなどは簡単であろうと，勝手に判断してしまうのである。**達者で暮らしていただきたい**，と言いつつ何の説明もせずに次に進んでしまうのである。

リスト2-8 ROM CG読み出しの機械語部分

```
                                  .Z80
                                  .PHASE  0E000H
                          ;
  E000     EB                     EX      DE,HL
  E001     46                     LD      B,(HL)   ;HI
  E002     23                     INC     HL
  E003     4E                     LD      C,(HL)   ;LOW
  E004     23                     INC     HL
  E005     ED 43 E077             LD      (BCWORK),BC
  E009     56                     LD      D,(HL)   ;COUNT
  E00A     23                     INC     HL
  E00B     5E                     LD      E,(HL)   ;ASCII CODE
  E00C     23                     INC     HL
  E00D     CD E017                CALL    SET0
                          ;
  E010     01 1400                LD      BC,1400H
  E013     CD E051                CALL    READCG
  E016     C9                     RET
                          ;
  E017     E5             SET0:   PUSH    HL       ;SAVE POINTER
  E018     D5                     PUSH    DE       ;SAVE COUNT and ASCII CO
DE
  E019     01 1FD0                LD      BC,1FD0H         ;FOR turbo
  E01C     AF                     XOR     A
  E01D     ED 79                  OUT     (C),A
                          ;
  E01F     ED 4B E077             LD      BC,(BCWORK)
  E023     21 3800                LD      HL,3800H         ;KANJI VRAM(turb
o)
  E026     3E 00                  LD      A,00H
  E028     CD E047                CALL    SET1
                          ;
  E02B     ED 4B E077             LD      BC,(BCWORK)
  E02F     21 2800                LD      HL,2800H         ;ATTRIBUTE
  E032     D1                     POP     DE
  E033     D5                     PUSH    DE
  E034     3E 00                  LD      A,00H            ;ROM CG,COLOR=0
  E036     CD E047                CALL    SET1
                          ;
  E039     ED 4B E077             LD      BC,(BCWORK)
  E03D     21 3000                LD      HL,3000H         ;VRAM
  E040     D1                     POP     DE
  E041     7B                     LD      A,E              ;ASCII CODE
  E042     CD E047                CALL    SET1
                          ;
  E045     E1                     POP     HL
  E046     C9                     RET
                          ;
  E047     09             SET1:   ADD     HL,BC
  E048     44                     LD      B,H
  E049     4D                     LD      C,L      ;BC=HL+BC
  E04A     ED 79          SET2:   OUT     (C),A
  E04C     03                     INC     BC       ;INC ADDRESS
  E04D     15                     DEC     D        ;DEC COUNTER
```

```
E04E     20 FA                     JR      NZ,SET2
E050     C9                        RET
                           ;
E051     1E 08             READCG: LD      E,08H    ;COUNTER
E053     D9                        EXX
                           ;
E054     F3                        DI
E055     01 1A01                   LD      BC,1A01H
E058     ED 78             VDISP0: IN      A,(C)
E05A     F2 E058                   JP      P,VDISP0
E05D     ED 78             VDISP1: IN      A,(C)
E05F     FA E05D                   JP      M,VDISP1
                           ;
E062     D9                        EXX
E063     ED 78             SETP:   IN      A,(C)    ;12,READ 1 BYTE
E065     77                        LD      (HL),A   ;7,STORE DATA
E066     23                        INC     HL       ;6,INC POINTER
E067     03                        INC     BC       ;6,INC I/O ADDRESS
                           ;7+12+6+6=31
                           ;
E068     00                        NOP              ;4
E069     23                        INC     HL       ;6
E06A     2B                        DEC     HL       ;6,DUMMY
                           ;4+6+6=16
                           ;
E06B     3E 0D                     LD      A,0DH    ;7
E06D     3D                DLY:    DEC     A        ;4
E06E     C2 E06D                   JP      NZ,DLY   ;10
E071     1D                        DEC     E        ;4
E072     C2 E063                   JP      NZ,SETP  ;10
                           ;7+(4+10)*13+4+10=203
                           ;
                           ;31+16+203=250
E075     FB                        EI
E076     C9                        RET
                           ;
E077                       BCWORK: DS      2
                                   END
```

リスト 2-9　ROM CG 読み出し

```
100 CLEAR &HE000
110 MEM$(&HE000,16)=HEXCHR$("EB 46 23 4E 23 ED 43 77 E0 56 23 5E 23 CD 17 E0")
120 MEM$(&HE010,16)=HEXCHR$("01 00 14 CD 51 E0 C9 E5 D5 01 D0 1F AF ED 79 ED")
130 MEM$(&HE020,16)=HEXCHR$("4B 77 E0 21 00 38 3E 00 CD 47 E0 ED 4B 77 E0 21")
140 MEM$(&HE030,16)=HEXCHR$("00 28 D1 D5 3E 00 CD 47 E0 ED 4B 77 E0 21 00 30")
150 MEM$(&HE040,16)=HEXCHR$("D1 7B CD 47 E0 E1 C9 09 44 4D ED 79 03 15 20 FA")
160 MEM$(&HE050,16)=HEXCHR$("C9 1E 08 D9 F3 01 01 1A ED 78 F2 58 E0 ED 78 FA")
170 MEM$(&HE060,16)=HEXCHR$("5D E0 D9 ED 78 77 23 03 00 23 2B 3E 0D 3D C2 6D")
180 MEM$(&HE070,16)=HEXCHR$("E0 1D C2 63 E0 FB C9 00 00 00 00 00 00 00 00 00")
190 DEFUSR0=&HE000
200 '
210 WIDTH80
220 C=0
230 A$=USR0(CHR$(&H7,&HD0,48,C)+STRING$(8,&H30))
240 A$=RIGHT$(A$,8)
250 PRINT #0 A$
260 PRINT #0 LEFT$(CGPAT$(C),8)
```

## turboにおけるPCG/CG

さあて，話は変わってturboである。ここまでの話からも分かるよーに，X1のPCGは強力無比と言えるのだが，PCG定義，読み出しの速度と手間は実にうっとーしいのである。伝え聞くところによると，まともにインターフェイスなどを作ったらコストがかかりすぎるので，あのよーになったということだが，そのウワサの真偽はともかくとして，うっとーしいということだけは真実なのである。シャープもその点については正しく理解していたらしく，turboでは「高速アクセスモード」という機能が付けられている。これ

は，CGアクセスの下準備をVRAM，漢字VRAM，アトリビュートに各1バイト，計3バイトを書き込むだけで済むようにしたうえに，垂直帰線期間だけでなく，水平帰線期間にもアクセス可能にしたものである。つまり，走査線1本ごとにアクセスできるわけである。しかもうれしいことにソフトウェアでタイミングを取る必要もないのである。どーしてかというと，turboがハード的に，勝手にタイミングを取ってくれるのだ。すなわち，PCGにデータをセットしようとして，CPUがOUT命令を実行したとすると，その瞬間にウエイトがかけられてタイミングが取れるのである。ウエイトとは平たく言えば「金縛り」みたいなもので，CPUは気絶させられるわけである。で，周りの回路が準備OKになったところでCPUに「活」を入れる。CPUは何も気付かずにOUT命令を完了して，また4MHzのステップを踏み続けるのである。

turboではこの高速アクセスモードにより，256個のPCGを，低解像度モード(200ライン）のとき約0.5秒，高解像度モード（400ライン）のとき約0.27秒に短縮できたのである。さもなくばSYMBOL文などは**大ひんしゅくであったろう**。めでたしめでたし。

まずは実物を見ていただきたい。**リスト2-10，2-11**である。リスト2-10はこの章のア

**リスト2-10　turbo用CG R/Wプログラムの機械語部分**

```
                                        .Z80
                                        .PHASE  0E000H
                              ;
    E000     EB                         EX      DE,HL
    E001     01 20D0                    LD      BC,1FD0H+100H
    E004     ED A3                      OUTI            ;DATA TO (1FD0H)
                              ;
    E006     01 40FF                    LD      BC,3FFFH+100H
    E009     ED A3                      OUTI            ;KANJI VRAM
                              ;
    E00B     01 28FF                    LD      BC,27FFH+100H
    E00E     ED A3                      OUTI            ;ATTRIBUTE
                              ;
    E010     01 38FF                    LD      BC,37FFH+100H
    E013     ED A3                      OUTI            ;VRAM
                              ;
    E015     56                         LD      D,(HL)  ;COUNT
    E016     23                         INC     HL
    E017     5E                         LD      E,(HL)  ;STEP
    E018     23                         INC     HL
    E019     46                         LD      B,(HL)  ;14H,15H,16H or 17H
    E01A     23                         INC     HL
    E01B     0E 00                      LD      C,00H
    E01D     7E                         LD      A,(HL)
    E01E     23                         INC     HL      ;HL POINTS DATA AREA
                              ;
    E01F     B7                         OR      A       ;A=0 THEN SETPCG ELSE RE
AD
    E020     C2 E02E                    JP      NZ,READ
                              ;
    E023     04               WRITE:    INC     B
    E024     ED A3                      OUTI
    E026     79                         LD      A,C
    E027     83                         ADD     A,E
    E028     4F                         LD      C,A     ;C=C+STEP
    E029     15                         DEC     D
    E02A     C2 E023                    JP      NZ,WRITE
    E02D     C9                         RET
                              ;
    E02E     ED A2            READ:     INI
    E030     04                         INC     B
    E031     79                         LD      A,C
    E032     83                         ADD     A,E
    E033     4F                         LD      C,A     ;C=C+STEP
    E034     15                         DEC     D
```

```
E035      C2 E02E                        JP      NZ,READ
E038      C9                             RET
                                    ;
                                         END
```

リスト 2-11　turbo 用 CG R/W プログラム

```
100 CLEAR &HE000
110 MEM$(&HE000,16)=HEXCHR$("EB 01 D0 20 ED A3 01 FF 40 ED A3 01 FF 28 ED A3")
120 MEM$(&HE010,16)=HEXCHR$("01 FF 38 ED A3 56 23 5E 23 46 23 0E 00 7E 23 B7")
130 MEM$(&HE020,16)=HEXCHR$("C2 2E E0 04 ED A3 79 83 4F 15 C2 23 E0 C9 ED A2")
140 MEM$(&HE030,16)=HEXCHR$("04 79 83 4F 15 C2 2E E0 C9 00 00 00 00 00 00 00")
150 DEFUSR0=&HE000
160 '
170 WIDTH 40
180 '1    2     3   4    5            6         7            8
190 '1FD0,KANJI,ATT,VRAM,COUNT(16/8),STEP(2/1),I/O PORT HIGH,R/W
200 '
210 '８＊８の　ＲＯＭ　ＣＧを読む（アルファベット）
220 A$=USR0(CHR$(&H23,0,0,66,8,2,&H14,1)+STRING$(48,0)):L=8:GOSUB"DUMP"
230 '８＊１６の　ＲＯＭ　ＣＧを読む（アルファベット）
240 A$=USR0(CHR$(&H63,0,0,66,16,1,&H14,1)+STRING$(48,0)):L=16:GOSUB"DUMP"
250 '８＊１６の　ＲＯＭ　ＣＧを読む（漢字：『筋』の左側）
260 A$=USR0(CHR$(&H63,&H86,0,&HDA,16,1,&H14,1)+STRING$(48,0)):L=16:GOSUB"DUMP"
270 SUJI$=A$
280 '
290 GOSUB"ALL"
300 '
310 'ＰＣＧをノーマルモードでセット
320   A$=USR0(CHR$(&H23,0,&H20,0,8,2,&H15,0)+STRING$(8,&H3))
330   A$=USR0(CHR$(&H23,0,&H20,0,8,2,&H16,0)+STRING$(8,&HC))
340   A$=USR0(CHR$(&H23,0,&H20,0,8,2,&H17,0)+STRING$(8,&H30))
350 '
360 'ＰＣＧを外字モードでセット
370   A$=USR0(CHR$(&H23,&H10,&H20,4,16,1,&H15,0)+SUJI$)
380   A$=USR0(CHR$(&H23,&H10,&H20,4,16,1,&H16,0)+SUJI$)
390   A$=USR0(CHR$(&H23,&H10,&H20,4,16,1,&H17,0)+SUJI$)
400 '
410 OUT &H31A4,&H4
420 OUT &H39A4,&H10
430 OUT &H21A4,&H27      :'左筋を表示
440 END
450 '
460 LABEL"DUMP"
470 A$=MID$(A$,9,L)
480 FOR I=1 TO L
490 B=ASC(MID$(A$,I,1)):B$=RIGHT$("0000000"+BIN$(B),8):PRINTB$
500 NEXT:PRINT
510 Q$=INKEY$(1)
520 RETURN
530 '
540 LABEL"ALL"
550 CLS:KMODE 0
560 CGEN 1:LOCATE 0,0
570 FOR I=0 TO 255:PRINT #0,CHR$(I);:NEXT
580 CGEN 0:KMODE 1
590 RETURN
```

センブラで書かれたプログラムの中では一番短いものだが，なんと turbo の PCG，ROM CG，漢字 ROM の読み書きすべてに対応してしまっているのだ(多少はナニだけどね)。リスト 2-10 の中のループの E028$_H$番地と E033$_H$番地で C レジスタを変化させているが，これはちゃんと意味があってやっている。使い方はリスト 2-11 の 180，190 行を見ていただきたい。パラメータは順に，

- 1FD0$_H$へのデータ
- 漢字 VRAM へのデータ
- アトリビュートへのデータ
- VRAM へのデータ

● COUNT（8×8を読むか8×16を読むか）
● STEP（8×8なら2，8×16なら1）
● I/Oポートの上位アドレス（$14_H$〜$17_H$）
● R/Wのフラグ（読む＝1／書く＝0）

となっている。

210行からは，まず8×8のROM CGの読み出しである。ASCIIコードは66＝〝B〞である。第1パラメータが$23_H$で，**図2-3**を見るとビット6＝0，ビット5＝1である。すなわち8ラスタCGアクセスで，高速アクセスである。おっと，ビット5の説明では「PCG」となっている。愛敬愛敬。さて，第6パラメータが2になっているが，なぜそのようになっているかというと，8×8の「B」のパターンは，表示されるときは

$7C_H$，$42_H$，……

となっているのだが，読み出すと

$7C_H$，$7C_H$，$42_H$，$42_H$，……

と2度ずつ繰り返してしまうのだ。だから「BC＝BC＋2」としているのだ。230行からはビット6＝1となり，8×16のCGに対するアクセスである。このビット6により，アクセスするROMが変わるのである(表示されるCGは変わらない)。250行からの漢字ROMの読み出しも同じようなものである。

次に310行から2度PCGをセットしている。turboでのPCGには，8×8として扱われるノーマルモードと，8×16として扱われる外字モードの二つのモードがある。外字モードの場合は偶数番目と，その次の奇数番目のPCG 2個が1個の8×16の文字として扱われる(たとえば0と1，2と3などの組)。よって，16×16の漢字は4個のPCGを合体させて作ることができるわけである。256÷4＝64で，turboは64個の漢字の外字を持てる

**図2-3　$1FD0_H$＝画面管理ポートの意味**

| データ内容 | コントロール |
|---|---|
| ビット 0 | 0＝低解像度モニタ(200ライン)<br>1＝高解像度モニタ(400ライン)　　モニタ切り換え |
| ビット 1 | 0＝1本ラスタ/ドット<br>1＝2本ラスタ/ドット |
| ビット 2 | 0＝ノーマル（8ラスタ/CHAR)(25行，20行)<br>1＝漢字　　(16ラスタ/CHAR)(12行，10行) |
| ビット 3 | 0＝バンク0表示<br>1＝バンク1表示 |
| ビット 4 | 0＝バンク0アクセス<br>1＝バンク1アクセス |
| ビット 5 | 0＝PCGコンパチアクセス<br>1＝PCG高速アクセス |
| ビット 6 | 0＝8ラスタCGアクセス<br>1＝16ラスタCGアクセス |
| ビット 7 | 0＝アンダーラインなし<br>1＝アンダーラインあり |

ことになる。410行からは「筋」の左半分を表示している。漢字VRAMのビット4が立っていて，外字モードになっている点に注意。これによりPCGの4番と5番が上下になって同時に表示されるわけである。

以上がPCGなわけであった。X1ではPCGを使うことによりALL BASICでも，まともに遊べるゲームができてしまうのである。マシン語にすればもっと遊べるのである。というわけで，PCGはゲーム街道をお通りなのであった。

# 第3章

# 漢字



漢字名野出亜留

# 第3章
# 漢字名野出亜留

第2章ではPCGをやったのである。となれば，この章では漢字ということになるわけだ。

## 漢字コードの基本

大体の人は知っているよーに，漢字にもASCIIコードなどと同じように，「漢字コード」なるものがあり，**整理整頓**に一役買っているのである。そいでもって，この漢字コードというやつは実に食わせものなのだ。まず，恐ろしいことに，この漢字コードには3種類が群雄割拠しているのである。

**① JIS漢字コード**

**② 区点コード**

**③ シフトJIS漢字コード**

である。これらは三つとも「2バイトコード」，つまり，2バイトで漢字1文字を表現するコードである。

①まず，JIS漢字コードの場合は，

上位バイト（1バイト目）は $21_H$〜$7E_H$

下位バイト（2バイト目）も $21_H$〜$7E_H$

となっている。ちなみに「鯖」のJISコードは$3B2A_H$である。

さて，それぞれのバイトで，$21_H$〜$7E_H$しか使っていないことからも分かるように，このコードには「隙間」ができている。たとえば，$0000_H$〜$2120_H$が空いてるし，$217F_H$〜$2220_H$なども空いているのである。なんでこうなっているかというと，どうやらコントロールコードを入れたくなかったからのようである。つまり，ASCIIコードでは$00_H$〜$20_H$は（$20_H$のスペースは違うけど）コントロールコードとして特殊な機能を割り当てられているから，避けたようなのである。また，それ以外にも空きがある。たとえば，$2577_H$〜$257E_H$などである。この部分はカタカナとギリシャ文字の隙間なのである。

そいでもって，JIS漢字コードは第1水準と第2水準に分かれるわけであるが，

第1水準＝$2121_H$〜$4F53_H$

第2水準＝$5020_H$〜$7424_H$

となっている。第1水準の$4F54_H$〜$4F7E_H$と，第2水準の$7425_H$〜$7E7E_H$は未定義（空き）である。

なぜ第1，第2水準に分けられているかというと，これは使用頻度によるわけだ。たとえば第2水準には國（国），學（学）などの旧字体や，**瑟竊鬘蠡髑黶黿**などの，見たことも聞いたこともないよーな漢字が並んでいるわけである。それはともかく，一言注意しておかなければならないのは，第1水準が（記号は別として）「読み」ごとに分類されているの

に対して，第2水準は，部首（偏や冠など）ごとに分類されているということである。第2水準の漢字には，読み方も分からないようなものが多いから，妥当な気もするが，**読み方も分からないよーな漢字をどう使うのであろうか。**ま，どーでもいいけどね。

②次に区点コードである。

区点コードはJISコードを少しアレンジしただけのものである。すなわち，JISの上位，下位それぞれから$20_H$を引いて，10進数で表し，もう1回くっつけ直したものである。だから，たとえばJISコードの$4B7A_H$（繭）は，

$4B_H-20_H=2B_H=43$

$7A_H-20_H=5A_H=90$

ということになり，区点コードは「4390」となるのである。はっきり言うが，これは4桁の10進数ではなく，「2桁の10進数が二つ」なのである。

この区点コードというやつは，ただ単に入力する際にA～Fを押さなくて済むというぐらいの意味しかない。これはまだ音訓変換さえもなかったような大昔に，漢字の入力方法として考え出されたものらしい。昔はコンピュータで漢字を使うのは大苦労だったのである。

③シフトJIS漢字コード，

このシフトJISというものは，JISコードの短所を補おうとしてできたものである（結果的には補えたのかどーだか知らない）。

たとえばJISコードの$4141_H$（疏）を，普通の方法でプリントしようとすると，$41_H$はASCIIコードでは「A」であるから，画面には「AA」と表示されることになる。そのよーなことになっては，どーしょーもないので，JISコードを使う場合にはどうしても「漢字IN/OUT」という，コントロールコードが必要になってくる。つまり，この場合だと，

（漢字IN），$41_H$，$41_H$，（漢字OUT）

の6バイトのデータで「疏」1文字を表現することになる。

それでは不便だということでできたのがシフトJISコードである。こいつは，「第1バイトが$81_H$～$9F_H$もしくは$E0_H$～$EF_H$で，なおかつ，第2バイトが$40_H$～$7E_H$もしくは$80_H$～$FC_H$の4桁の16進数を漢字とみなす」というものである。ダサイことに第1バイトの範囲に隙間がある。これは，半角のカタカナのASCIIコード$A0_H$～$DF_H$だからなのである。シフトJISを考え出した人は，半角のカタカナと共存させたかったようである。もっとも，カタカナの代わりにグラフィックキャラクタの一部が使えなくなる。しかし，普通の文字（半角のアルファベットなど）と漢字をほとんど同じように扱えるというメリットが出てくるわけである。

ちなみに先程の「疏」は，シフトJISコードだと，

$9160_H$

となり，「$91_H$，$60_H$」の2バイトのデータだけで「疏」を表現できることになる。第1バイトが$91_H$であることが，漢字INの代わりになり，第2バイトの次に暗黙の漢字OUTがあるわけだ。つまり，これによって「漢字IN/OUT」が不要になる。

そこでこの三つにコードの間の変換プログラムが必要となってしまうわけだ。というと

ところで，

**リスト 3-1**：JIS ↔シフト JIS

**リスト 3-2**：JIS ↔区点

の変換プログラムとなっている。

リスト 3-1　JIS↔シフトJIS

```
100 DEFSNG J
110 FOR J=&H8100 TO &H9FFF:GOSUB150:NEXT:'すべての漢字を走る
120 FOR J=&HE000 TO &HEB00:GOSUB150:NEXT
130 END
140 '
150 LOCATE0,0:PRINTCHR$(J):P$=SCRN$(0,0,2):'一度表示してから
160 JI$=JIS$(P$):GOSUB 260                   :'変換し
170 IF EC THEN STOP
180 IF SJ$<>HEX$(ASC(P$)) THEN STOP          :'チェックする
190 '
200 SJ$=HEX$(ASC(P$)):GOSUB 420              :'変換し
210 IF EC THEN STOP
220 IF JI$<>JIS$(P$) THEN STOP               :'チェックする
230 RETURN
240 '
250 'JIS->SHIFT JIS
260 EC=0
270 W$=MKI$(VAL("&H"+JI$))
280 JI1=ASC(RIGHT$(W$,1)):JI2=ASC(LEFT$(W$,1))
290 '
300 IF (JI1<&H21) OR (&H7E<JI1) THEN EC=1:RETURN
310 IF (JI2<&H21) OR (&H7E<JI2) THEN EC=1:RETURN
320 SJ1=INT((JI1-&H21)/2)+&H81
330 IF JI1>=&H5F THEN SJ1=SJ1+(&HE0-&H9F-1)
340 '
350 IF JI1 AND 1 THEN SJ2=&H40:GOTO 360 ELSE SJ2=&H9F:GOTO 370
360 IF (&H60<=JI2) THEN SJ2=SJ2+1
370 SJ2=SJ2+JI2-&H21
380 SJ=SJ1*256+SJ2:SJ$=HEX$(SJ)
390 RETURN
400 '
410 'SHIFT JIS->JIS
420 EC=0
430 W$=MKI$(VAL("&H"+SJ$))
440 SJ1=ASC(RIGHT$(W$,1)):SJ2=ASC(LEFT$(W$,1))
450 '
460 IF (&H81<=SJ1) AND (SJ1<=&H9F) THEN 490
470 IF (&HE0<=SJ1) AND (SJ1<=&HEF) THEN SJ1=SJ1-&HE0+&H9F+1:GOTO 490
480 EC=1:RETURN
490 JI1=INT((SJ1-&H81)*2)+&H21
500 '
510 IF (&H40<=SJ2) AND (SJ2<=&H7E) THEN 540
520 IF (&H80<=SJ2) AND (SJ2<=&HFC) THEN SJ2=SJ2-1:GOTO 540
530 EC=1:RETURN
540 IF SJ2>=&H9E THEN JI1=JI1+1:SJ2=SJ2-&H9E+&H40
550 JI2=SJ2-&H40+&H21
560 '
570 JI$=HEX$(JI1)+HEX$(JI2)
580 RETURN
```

リスト 3-1 は，JI$に JIS コードを入れ，GOSUB 260 とすると，SJ$にシフト JIS コードが入って返ってくる。その逆が GOSUB 420 である。ただし，どちらの場合でも，EC=1でリターンしたなら，あり得ないコードを変換しようとしたということである（つまりエラー）。

**リスト 3-2　JIS↔区点**

```
100 DEFSNG J
110 FOR J=&H8100 TO &H9FFF:GOSUB150:NEXT
120 FOR J=&HE000 TO &HEB00:GOSUB150:NEXT
130 END
140 '
150 LOCATE0,0:PRINT CHR$(J):P$=SCRN$(0,0,2)
160 JI$=JIS$(P$):GOSUB 240
170 IF KT$<>KTN$(P$) THEN STOP
180 '
190 KT$=KTN$(P$):GOSUB 320
200 IF JI$<>JIS$(P$) THEN STOP
210 RETURN
220 '
230 'JIS->KUTEN
240 W$=MKI$(VAL("&H"+JI$))
250 JI1=ASC(RIGHT$(W$,1)):JI2=ASC(LEFT$(W$,1))
260 KT1=JI1-&H20
270 KT2=JI2-&H20:'単にバイアスをかけるだけ
280 KT$=RIGHT$(STR$(10000+KT1*100+KT2),4)
290 RETURN
300 '
310 'KUTEN->JIS
320 W=VAL(KT$)
330 KT1=W ¥ 100:KT2=W MOD 100
340 JI1=KT1+&H20
350 JI2=KT2+&H20:'単にバイアスをかけるだけ
360 JI$=HEX$(JI1*256+JI2)
370 RETURN
```

リスト 3-2 は SJ$でなく，KT$（区点コード）になっただけである。

細かなプログラムの動きは，**図 3-1** の「コード相関図」を見て納得していただきたいと思うしだいである。

**図 3-1　コード相関図**

Ⓐ シフトJISコードとJIS漢字コードの関係

| mm \ nn | 00　3F | 40　7E | 7F | 80　9E | 9F　FC | FF |
|---|---|---|---|---|---|---|
| 00 | | | | | | |
| 81 | | 2121　215F<br>2321　235F | | 2160　217E<br>2360　237E | 2221　277E<br>2421　277E | |
| 9F | | 5D21　5D5F | | 5D60　5D7E | 5E21　5E7E | |
| | | | | | | |
| E0 | | 5F21　5F5F | | 5F60　5F7E | 6021　607E | |
| EF | | 7D21　7D5F | | 7D60　7D7E | 7E21　7E7E | |
| FC | | | | | | |
| FF | | | | | | |

mm，nnはそれぞれシフトJISコードの第1，第2バイトを示す
▒は未使用領域

Ⓑ シフトJISコードと区点コードの関係

| mm＼nn | 00 3F | 40 7E | 7F | 80 9E | 9F FC | FF |
|---|---|---|---|---|---|---|
| 00 | | | | | | |
| 81 | | 0101 0163<br>0301 0363 | | 0164 0194<br>0364 0394 | 0201 0294<br>0401 0494 | |
| 9F | | 6101 6163 | | 6164 6194 | 6201 6294 | |
| | | | | | | |
| E0 | | 6301 6363 | | 6364 6394 | 6401 6494 | |
| EF | | 9301 9363 | | 9364 9394 | 9401 9494 | |
| FC | | | | | | |
| FF | | | | | | |

mm, nnはそれぞれシフトJISコードの第1, 第2バイトを示す
□は未使用領域

# 表示するのである

X1/turbo で漢字を表示するには,

① **漢字 VRAM を使う（turbo）**

② **グラフィック画面に描く**

の2とおりの方法があるわけだ。①の場合は，turbo の漢字 VRAM に適当な値を**シュポポポン**と数バイト書き込むだけ——などと思ったら大間違い。漢字 VRAM に書き込むべきデータ（漢字 ROM アドレスという）の計算が結構面倒なのである。

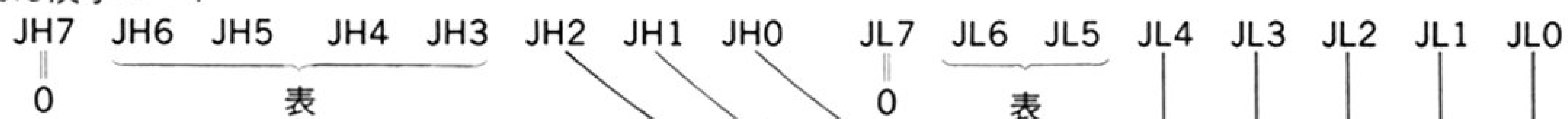
まず，JIS コードは2バイトであるから，すなわち16ビットなわけだ。しかし，よーく見ると，第1，第2バイトともに，$21_H$(&B00100001)〜$7E_H$(&B01111110)の範囲である。すなわち，どちらのバイトでも，**第7ビット**が0なのだ。よって，JIS コードは実際のところ7×2＝14ビットで済んでしまうということになる。それに対して，turbo の漢字 ROM アドレスというのは，I/O マップの漢字 VRAM の所を見てもらえば分かるが，13ビット（8＋5ビット）の値なのである。まだ1ビットの差が残っているが，これはいかなるわけかというと，JIS コードの $2821_H$〜$2F7E_H$と，$7425_H$〜$7E7E_H$が空いているからなどの理由によるのである(ここらへんのところは，JIS の改訂などがあるのでちょいと違う部分もある)。

というわけで，**図 3-2** が JIS コード↔漢字 ROM アドレス変換の方法を図示したもの，**リスト 3-3** が変換を実行するプログラムである。少し付け加えておくが，要するに，漢字 ROM アドレスの下位8ビット（$3000_H$〜$37FF_H$の I/O 空間に OUT するやつ）は，単

純に決定するのだが，3800$_H$～3FFF$_H$の漢字 VRAM の方のデータは図 3-2 の下の表を見なければならないということである。注意深く見れば規則性が見えてくるであろう。

**図 3-2　turbo のJIS コード↔漢字 ROM アドレスの関係**

●JIS漢字コード




●漢字ROMアドレス




| | JIS漢字コード | | | | | | 漢字ROMアドレス | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | JH6 | JH5 | JH4 | JL6 | JL5 | | RH4 | RH3 | RH2 | RH1 | RH0 | JIS漢字コード |
| 第1水準 | 0 | 1 | 0 | 0 | 1 | ←→ | 0 | 0 | 0 | 0 | JH3 | 2021$_H$～277E$_H$ |
| | 0 | 1 | 0 | 1 | 0 | ←→ | 0 | 0 | 0 | 1 | JH3 | |
| | 0 | 1 | 0 | 1 | 1 | ←→ | 0 | 0 | 0 | 0 | JH3 | |
| | 0 | 1 | 1 | 0 | 1 | ←→ | 0 | 0 | 1 | 0 | JH3 | 3021$_H$～3F7E$_H$ |
| | 0 | 1 | 1 | 1 | 0 | ←→ | 0 | 0 | 1 | 1 | JH3 | |
| | 0 | 1 | 1 | 1 | 1 | ←→ | 0 | 1 | 0 | 0 | JH3 | |
| | 1 | 0 | 0 | 0 | 1 | ←→ | 0 | 1 | 0 | 1 | JH3 | 4021$_H$～4F7E$_H$ |
| | 1 | 0 | 0 | 1 | 0 | ←→ | 0 | 1 | 1 | 0 | JH3 | |
| | 1 | 0 | 0 | 1 | 1 | ←→ | 0 | 1 | 1 | 1 | JH3 | |
| | | | | | | | | | | | | |
| 第2水準 | 1 | 0 | 1 | 0 | 1 | ←→ | 1 | 0 | 0 | 0 | JH3 | 5021$_H$～5F7E$_H$ |
| | 1 | 0 | 1 | 1 | 0 | ←→ | 1 | 0 | 0 | 1 | JH3 | |
| | 1 | 0 | 1 | 1 | 1 | ←→ | 1 | 0 | 1 | 0 | JH3 | |
| | 1 | 1 | 0 | 0 | 1 | ←→ | 1 | 0 | 1 | 1 | JH3 | 6021$_H$～6F7E$_H$ |
| | 1 | 1 | 0 | 1 | 0 | ←→ | 1 | 1 | 0 | 0 | JH3 | |
| | 1 | 1 | 0 | 1 | 1 | ←→ | 1 | 1 | 0 | 1 | JH3 | |
| | 1 | 1 | 1 | 0 | 1 | ←→ | 1 | 1 | 1 | 0 | JH3 | 7021$_H$～777E$_H$ |
| | 1 | 1 | 1 | 1 | 0 | ←→ | 1 | 1 | 1 | 1 | JH3 | |
| | 1 | 1 | 1 | 1 | 1 | ←→ | 1 | 1 | 1 | 0 | JH3 | |

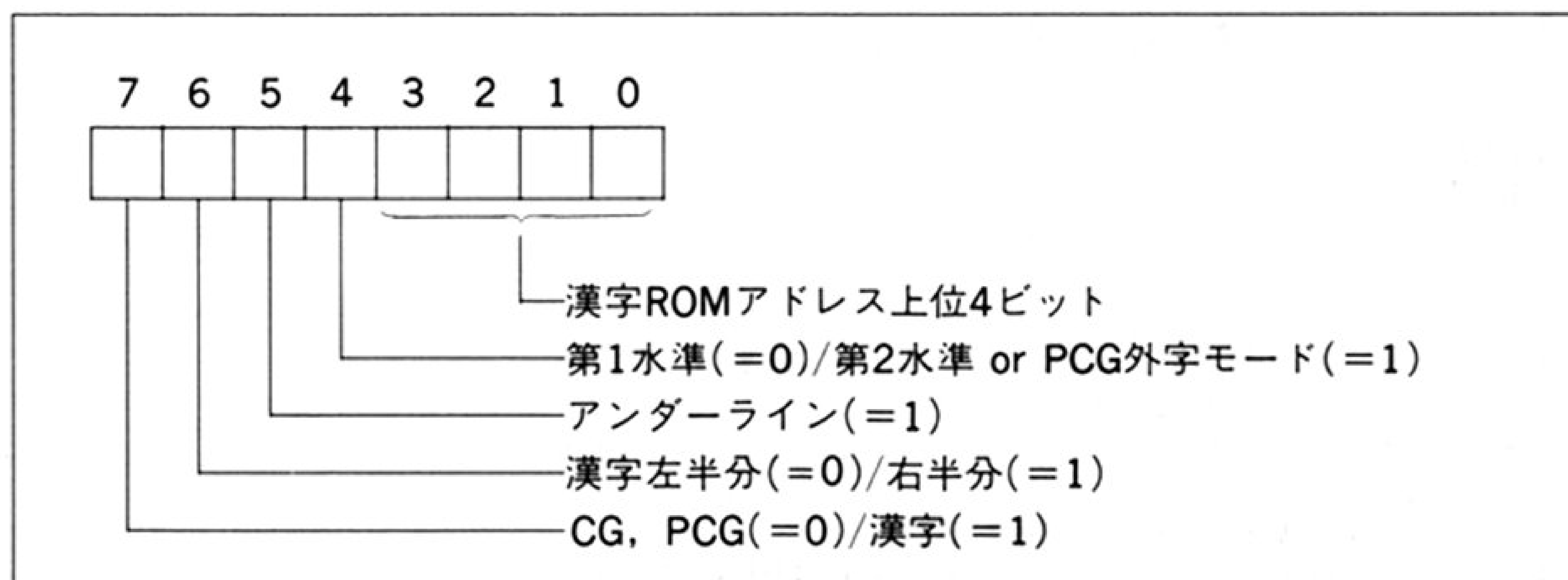
**リスト 3-3　turbo 用 JIS↔漢字 ROM アドレス変換**

```
100 DEFSNG J
110 FOR J=&H8100 TO &H9FFF:GOSUB 150:NEXT
120 FOR J=&HE000 TO &HEAFF:GOSUB 150:NEXT
130 END
140 '
150 LOCATE 0,0:PRINT CHR$(J):J$=SCRN$(0,0,2)
160 IF J$="※" THEN 290
170 JI$=JIS$(J$):GOSUB 320       :'JIS -> ROM
180 IF EC THEN STOP
190 IF RL<>INP(&H3000) THEN STOP
200 IF RH<>(INP(&H3800) AND &H1F) THEN STOP
210 OUT &H3804,RH OR &H80
220 OUT &H3805,RH OR &HC0
230 OUT &H3004,RL:OUT &H3005,RL
```

```
240 '
250 RL=INP(&H3000)
260 RH=INP(&H3800) AND &H1F
270 JI$="":GOSUB 490            :'ROM -> JIS
280 IF JI$<>JIS$(J$) THEN STOP
290 RETURN
300 '
310 'JIS->ROM
320 EC=0
330 JI=VAL("&H"+JI$)
340 RL=(JI AND &B11111) OR ((JI ¥ 8) AND &B11100000)
350 RW1=(JI ¥ &H1000) AND 7     :'JH6,JH5,JH4
360 RW2=(JI ¥ &H800)  AND 1     :'JH3
370 RW3=(JI ¥ &H20)   AND 3     :'JL6,JL5
380 ON RW1 GOTO 390,400,450,450,450,450,410
390 EC=1:RETURN
400 RH=&B0 OR RW2     :GOTO 420
410 RH=&B11100 OR RW2
420 IF RW3=2 THEN RH=RH OR 2
430 IF RW3=3 THEN RH=RH XOR 1
440 RETURN
450 RH=((RW1-3)*3+1+RW3)*2 OR RW2
460 RETURN
470 '
480 'ROM->JIS
490 EC=0
500 JI2=RL AND &B11111            :'RL4 - RL0
510 JI1=(RL ¥ &H20) AND &B111    :'RL7 - RL5
520 IF RH<4 THEN 580             :'2121H - 277EH
530 IF RH>&H1C THEN 590          :'7021H - 777EH
540 JI1=JI1 OR ((((RH ¥ 2)+7)¥3)*&H10)
550 JI1=JI1 OR (RH AND 1)*8
560 JI2=JI2 OR  (((((RH ¥ 2)+1)MOD3)+1)*&H20)
570 GOTO 650
580 JI1=JI1 OR &B100000 :GOTO 600
590 JI1=JI1 OR &B1110000:GOTO 600
600 RW=RH AND 3
610 IF RW=0 THEN JI2=JI2 OR &B100000:GOTO 650
620 IF RW=2 THEN JI2=JI2 OR &B1000000:GOTO 650
630 IF RW=1 THEN JI2=JI2 OR &B1100000:GOTO 650
640 EC=1:RETURN
650 JI$=HEX$(JI1)+HEX$(JI2):RETURN
```

210〜230行のルーチンは変換を実行した後で，実際に漢字VRAMへそのデータをOUTするものである。**図3-3**にあるように，左側/右側の指定なども行なっていることに注意である。

**図3-3　漢字VRAMの各ビットの機能**



# さらに表示するのである

②のグラフィックに表示する場合をやるのである。これは X1 と turbo の場合に分かれるのであるが，turbo の場合は，第 2 章でやった「CG（漢字 ROM）の読み出し」を実行した後で，そのデータをグラフィックに書き込んでしまえばよいわけであるから，さっさと X1 の場合に行ってしまうわけである。

X1 における漢字 ROM といえば，CZ-8KR，CZ-8BK2 である（X1F/G の model 20 には標準で内蔵している）。で，そいつのアクセス方法が主題になるわけだったりする。

漢字 ROM の読み出しは**リスト 3-4** である。これは JIS コードを入力して，対応する漢字を画面の左上に表示するプログラムである。説明すると，

120〜140 行：JIS コードを入力し，1 バイト目（JI1）と 2 バイト目（JI2）に分離する。

150〜170 行：ROM に JI 1 と 0 を OUT し，2 バイト読む。

**リスト 3-4　X1 用漢字 ROM 読み出しプログラム**

```
100 INIT:PALET 1,7:CLS4
110 '
120 LOCATE 0,5:INPUT JI$          :'INPUT JIS CODE
130 JI1=VAL("&H"+LEFT$(JI$,2))    :'JIS HIGH
140 JI2=VAL("&H"+RIGHT$(JI$,2))   :'JIS LOW
150 OUT &HE80,JI1
160 OUT &HE81,0
170 GOSUB 330     :'READ 2 BYTE FROM ROM
180               :'GET R1 (FROM TABLE)
190 '
200 R=R1*&H100+(JI2-&H20)*&H10
210 OUT &HE82,0
220 OUT &HE80,ASC(LEFT$(MKI$(R),1)):'SET LOW ADDR.
230 OUT &HE81,INT(R/256)           :'SET HIGH ADDR.
240 FOR I=0 TO 15
250   GOSUB 330 :'READ 2 BYTE FROM ROM
260   LP=R1:RP=R2
270   IF I<8 THEN AD=&H4000+I*&H800 ELSE AD=&H4050+(I-8)*&H800
280   OUT AD,LP:OUT AD+1,RP       :'DRAW PATTERN
290 NEXT
300 GOTO 120     :'LOOP
310 '
320 'READ FROM ROM
330 OUT &HE82,1
340 R1=INP(&HE80)
350 R2=INP(&HE81)
360 OUT &HE82,0
370 RETURN
```

これにより，「漢字 ROM 内のテーブル」を使って，本当の「漢字パターンの ROM 内アドレス」が分かるのである。これについては，**表 3-1** を見ていただきたい。これは参考文献 1 からの引用である。

結局そのようにして得られたROM内アドレス（200行の変数R）を，もう一度$E80_H$，$E81_H$番地に指定し，漢字パターンの読み出しにかかるのが，210〜290 行である。一度に読み出すのは 2 バイトで，それが 16×16 の漢字の横 1 ラインになる。結局それを 16 回繰り返すことによって，めでたく一つの漢字のパターンとなるのである。なお，270 行，280 行

表 3-1　漢字 ROM (CZ-8KR, 8KR2)の I/O ポート

| I/Oアドレス | 入出力 | 機　　　　能 |
|---|---|---|
| 0E82H | 出　力 | 00H出力……ROMからのデータ読み出し終了<br>01H出力……ROMからのデータ読み出し開始 |
| 0E81H<br>0E80H | 入出力 | I/O(0E81H)←00H<br>I/O(0E80H)←JIS漢字コード上位バイト　}のとき<br>データ読み出し時には,<br>I/O(0E81H)からは00H<br>I/O(0E80H)からはROM内アドレス上位バイト<br>が得られる。 |
| | | I/O(0E81H)←ROM内アドレス上位バイト(≒00H)<br>I/O(0E80H)←ROM内アドレス下位バイト　}のとき<br>データ読み出し時には,<br>I/O(0E81H)からは，右部分フォントパターン16バイト分<br>I/O(0E80H)からは，左部分フォントパターン16バイト分<br>が得られる。 |

は簡便にグラフィックの漢字を表示するためのルーチンである。

ところで，先に言っておくが，機械語で漢字 ROM にアクセスする場合は，E82Hへ 01H を OUT してからデータを読み始めるまでの間（リスト 3-4 の 335 行にあたる）に 3 μs 以上ウエイトが必要ということになっている（リスト 3-5 参照)。

というところで，駄目押しとして，漢字 ROM にアクセスする機械語プログラムが**リスト 3-5** で，それを BASIC から使っているのが**リスト 3-6** である。当然なことであるが turbo では動かない。

リスト 3-5　機械語で漢字 ROM にアクセス

```
                                  .Z80
                                  .PHASE   0FE00H
                             ;
                             ;KANJI PATTERN READ
                             ;DE -> JIS-HIGH,JIS-LOW
 FE00     EB                 START1: EX        DE,HL
                             ;
 FE01     56                         LD        D,(HL)
 FE02     23                         INC       HL
 FE03     5E                         LD        E,(HL)
 FE04     18 0F                      JR        KREAD
                             ;
                             ;========================
                             ;KANJI PATTERN DRAW
                             ;DE -> JIS-HIGH,JIS-LOW,X,Y
                             ;!!! X,Y = 16 bits !!!
 FE06     EB                 START2: EX        DE,HL
                             ;
 FE07     56                         LD        D,(HL)
 FE08     23                         INC       HL
 FE09     5E                         LD        E,(HL)
 FE0A     23                         INC       HL
                             ;
 FE0B     4E                         LD        C,(HL)
 FE0C     23                         INC       HL
 FE0D     46                         LD        B,(HL)
 FE0E     23                         INC       HL        ;GET X
                             ;
 FE0F     7E                         LD        A,(HL)
 FE10     23                         INC       HL
 FE11     66                         LD        H,(HL)    ;GET Y
 FE12     6F                         LD        L,A
```

```
FE13    18 16             ;
                                  JR      KDRAW
                          ;
                          ;**************************
                          ;DE=JIS CODE:D=HIGH,E=LOW
                          ;READ KANJI PATTERN
                          ;
FE15    CD FE6F           KREAD:  CALL    SETROM
                          ;
FE18    21 FED0                   LD      HL,PATA ;PATTERN AREA
FE1B    3E 10                     LD      A,16
FE1D    08                KREADL: EX      AF,AF'  ;SAVE COUNTER
FE1E    CD FE8F                   CALL    RROM
FE21    72                        LD      (HL),D  ;GET LEFT
FE22    23                        INC     HL
FE23    73                        LD      (HL),E  ;GET RIGHT
FE24    23                        INC     HL
FE25    08                        EX      AF,AF'  ;CHECK COUNT
FE26    3D                        DEC     A
FE27    C2 FE1D                   JP      NZ,KREADL
FE2A    C9                        RET
                          ;
                          ;DE=JIS CODE:D=HIGH,E=LOW
                          ;BC=X,HL=Y
                          ;DRAW KANJI PATTERN
                          ;
FE2B    D5                KDRAW:  PUSH    DE      ;SAVE JIS CODE
FE2C    CD FEA3                   CALL    XYADDR  ;GET VRAM
FE2F    44                        LD      B,H
FE30    4D                        LD      C,L     ;COPY ADDR.
                          ;BC=G.ADDR.
FE31    D9                        EXX
FE32    D1                        POP     DE      ;GET JIS
FE33    CD FE6F                   CALL    SETROM
                          ;
FE36    01 0E82                   LD      BC,0E82H
FE39    D9                        EXX
                          ;
FE3A    AF                        XOR     A
FE3B    08                        EX      AF,AF'
FE3C    3E 10                     LD      A,16    ;COUNTER
                          ;
FE3E    08                KDRAWL: EX      AF,AF'
FE3F    D9                        EXX
                          ;BC=0E82H,A=0
FE40    3C                        INC     A
FE41    ED 79                     OUT     (C),A   ;BEGIN READ
FE43    0D                        DEC     C
FE44    0D                        DEC     C
FE45    00                        NOP             ;3 us
                          ;
FE46    ED 78                     IN      A,(C)   ;LEFT
FE48    0C                        INC     C
FE49    D9                        EXX
FE4A    ED 79                     OUT     (C),A   ;DRAW LEFT
FE4C    03                        INC     BC
FE4D    D9                        EXX
                          ;
FE4E    ED 78                     IN      A,(C)   ;RIGHT
FE50    D9                        EXX
FE51    ED 79                     OUT     (C),A   ;DRAW RIGHT
FE53    D9                        EXX
                          ;
FE54    AF                        XOR     A
FE55    0C                        INC     C
FE56    ED 79                     OUT     (C),A   ;END READ
                          ;BC=0E82H,A=0
FE58    D9                        EXX
                          ;BC=G.ADDR.
                          ;DOWN 1 LINE
FE59    0B                        DEC     BC      ;BACK ADDR
FE5A    21 0800                   LD      HL,800H
FE5D    B7                        OR      A
FE5E    ED 4A                     ADC     HL,BC   ;CHECK SIGN FLAG
FE60    F2 FE67                   JP      P,OKDN  ;OK(ONLY BLUE)
                          ;
FE63    21 C850                   LD      HL,4050H-7800H
                          ;IF WIDTH 40
                          ;       LD      HL,4028H-7800H
FE66    09                        ADD     HL,BC
FE67    44                OKDN:   LD      B,H
```

```
FE68    4D                          LD      C,L
                            ;
FE69    08                          EX      AF,AF'
FE6A    3D                          DEC     A
FE6B    C2 FE3E                     JP      NZ,KDRAWL
                            ;
FE6E    C9                          RET
                            ;
                            ;==============================
                            ;DE=JIS
FE6F    7B                  SETROM: LD      A,E     ;LOW
FE70    D6 20                       SUB     20H
FE72    6F                          LD      L,A     ;L=LOW-20H
                            ;
FE73    01 0E80                     LD      BC,0E80H
FE76    ED 51                       OUT     (C),D   ;E80 <- JISH
FE78    0C                          INC     C
FE79    AF                          XOR     A
FE7A    ED 79                       OUT     (C),A   ;E81 <- 00H
                            ;
FE7C    CD FE8F                     CALL    RROM    ;GET FROM TABLE
                            ;DE=TABLE VALUE
                            ;
FE7F    26 00                       LD      H,00H
                            ;HL=JIS LOW-20H='TEN'
                            ;
FE81    29                          ADD     HL,HL
FE82    29                          ADD     HL,HL
FE83    29                          ADD     HL,HL
FE84    29                          ADD     HL,HL   ;HL=HL*16
FE85    19                          ADD     HL,DE
                            ;NOW HL=ROM ADDRESS
                            ;
FE86    01 0E80                     LD      BC,0E80H
FE89    ED 69                       OUT     (C),L   ;E80 <- LOW
FE8B    0C                          INC     C
FE8C    ED 61                       OUT     (C),H   ;E81 <- HIGH
FE8E    C9                          RET
                            ;
                            ;READ ROM (JUST 2 BYTES)
FE8F    01 0E82             RROM:   LD      BC,0E82H
FE92    3E 01                       LD      A,1
FE94    ED 79                       OUT     (C),A   ;BEGIN READ
FE96    0D                          DEC     C
FE97    0D                          DEC     C
FE98    00                          NOP             ;3 us
                            ;
FE99    ED 50                       IN      D,(C)   ;LEFT
FE9B    0C                          INC     C
FE9C    ED 58                       IN      E,(C)   ;RIGHT
FE9E    AF                          XOR     A
FE9F    0C                          INC     C
FEA0    ED 79                       OUT     (C),A   ;END READ
FEA2    C9                          RET
                            ;
                            ;
                            ;ADDR=4000H+(X1>>3)+((Y1 & 7)<<11) +(Y1>>3)*80
                            ;ADDR=4000H+(BC/8) +((L AND 7)<<11)+(HL/8) *80
                            ;HL=Y1,BC=X1,BREAKS HL,A,BC,DE
                            ;return HL=addr,A=mask,D=count
                            ;
FEA3    7D                  XYADDR: LD      A,L     ;SAVE L
FEA4    CD FEC3                     CALL    DIV8    ;HL=HL/8
                            ;
FEA7    54                          LD      D,H
FEA8    5D                          LD      E,L     ;DE=HL
                            ;
FEA9    29                          ADD     HL,HL   ;
FEAA    29                          ADD     HL,HL   ;80=16*5
FEAB    19                          ADD     HL,DE   ;HL=5
FEAC    29                          ADD     HL,HL   ;10
FEAD    29                          ADD     HL,HL   ;20
FEAE    29                          ADD     HL,HL   ;40
FEAF    29                          ADD     HL,HL   ;80      ;HL=(HL/8)*80
                            ;OR     NOP                      (WIDTH 40)
                            ;
FEB0    E6 07                       AND     07H     ;A=(L AND 7)
FEB2    87                          ADD     A,A
FEB3    87                          ADD     A,A
FEB4    87                          ADD     A,A     ;A=((L AND 7)<<3)
                            ;
```

```
FEB5     C6 40                     ADD     A,040H   ;ADD 4000H
FEB7     57                        LD      D,A
FEB8     1E 00                     LD      E,00H
                          ;DE=4000H+((L AND 7)<<(3+8))
FEBA     19                        ADD     HL,DE    ;LAST 2 & 1ST WERE DONE
FEBB     EB                        EX      DE,HL    ;HDE=HL (SAVE)
                          ;
FEBC     60                        LD      H,B
FEBD     69                        LD      L,C      ;HL=BC
                          ;
FEBE     CD FEC3                   CALL    DIV8     ;HL=HL/8
FEC1     19                        ADD     HL,DE    ;BADDR DONE
FEC2     C9                        RET
                          ;HL has result (address),
                          ;
FEC3     CB 3C            DIV8:    SRL     H
FEC5     CB 1D                     RR      L
FEC7     CB 3C                     SRL     H
FEC9     CB 1D                     RR      L
FECB     CB 3C                     SRL     H
FECD     CB 1D                     RR      L
FECF     C9                        RET
                          ;
FED0                      PATA:    DS      32       ;PAT AREA
                          ;
                                   END
```

リスト 3-6　BASIC から「リスト 3-5」を使う

```
100 CLEAR &HFE00
110 MEM$(&HFE00,16)=HEXCHR$("EB 56 23 5E 18 0F EB 56 23 5E 23 4E 23 46 23 7E")
120 MEM$(&HFE10,16)=HEXCHR$("23 66 6F 18 16 CD 6F FE 21 D0 FE 3E 10 08 CD 8F")
130 MEM$(&HFE20,16)=HEXCHR$("FE 72 23 73 23 08 3D C2 1D FE C9 D5 CD A3 FE 44")
140 MEM$(&HFE30,16)=HEXCHR$("4D D9 D1 CD 6F FE 01 82 0E D9 AF 08 3E 10 08 D9")
150 MEM$(&HFE40,16)=HEXCHR$("3C ED 79 0D 0D 00 ED 78 0C D9 ED 79 03 D9 ED 78")
160 MEM$(&HFE50,16)=HEXCHR$("D9 ED 79 D9 AF 0C ED 79 D9 0B 21 00 08 B7 ED 4A")
170 MEM$(&HFE60,16)=HEXCHR$("F2 67 FE 21 50 C8 09 44 4D 08 3D C2 3E FE C9 7B")
180 MEM$(&HFE70,16)=HEXCHR$("D6 20 6F 01 80 0E ED 51 0C AF ED 79 CD 8F FE 26")
190 MEM$(&HFE80,16)=HEXCHR$("00 29 29 29 29 19 01 80 0E ED 69 0C ED 61 C9 01")
200 MEM$(&HFE90,16)=HEXCHR$("82 0E 3E 01 ED 79 0D 0D 00 ED 50 0C ED 58 AF 0C")
210 MEM$(&HFEA0,16)=HEXCHR$("ED 79 C9 7D CD C3 FE 54 5D 29 29 19 29 29 29 29")
220 MEM$(&HFEB0,16)=HEXCHR$("E6 07 87 87 87 C6 40 57 1E 00 19 EB 60 69 CD C3")
230 MEM$(&HFEC0,16)=HEXCHR$("FE 19 C9 CB 3C CB 1D CB 3C CB 1D CB 3C CB 1D C9")
240 '
250 WIDTH80:INIT:CLS 4
260 PALET 1,7
270 DEFUSR0=&HFE00
280 DEFUSR1=&HFE06
290 'D$=USR0(HEXCHR$("3E22"))
300 'D$=USR1(HEXCHR$("3E22")+MKI$(10)+MKI$(10))
310 '
320 X=0:Y=0
330 LOCATE 0,0:INPUT "JIS";JIS$
340 IF LEN(JIS$)<>4 THEN 330
350 D$=USR1(HEXCHR$(JIS$)+MKI$(X)+MKI$(Y))
360 X=X+16:IF X>640-16 THEN X=0:Y=Y+16:IF Y>200-16 THEN STOP
370 GOTO 330
```

リスト 3-5 の中には二つのプログラムが入っている。FE00$_H$番地からの START1 は漢字のパターンを，FEDA$_H$番地からの 32 バイトの領域に読み出すサブルーチンである。それに対して，FE06$_H$番地からの START2 は指定された位置（グラフィック）に，漢字を描くルーチンである。このルーチンは手抜きをしてあり，描画は青画面にしかしないようになっている。また，普通に考えるなら，「一旦読み出してから描く」のだろうが，それだと面白くないので，「ROM から読み出しながら描く」方法を取っている。

FE00$_H$〜からのルーチンの使い方は，DE レジスタが指しているアドレスから，JIS コードの上位，下位に入っていることが条件である。FE06$_H$〜は，さらにその後に X，Y の座標値が入っている必要がある。で，このようにすると BASIC から USR 関数でお手軽に使え

るというわけである。おっと念のために言っておくが，座標として指定するX，YはZ80の16ビット（2バイト）値で，下位，上位の順になっているわけだ。そして，Xの方は，8ドットごとになっている。つまり，（9，0）でも（10，0）でも，（8，0）に表示されてしまうのである。いちいちビットシフトなんかやっていられないのである。

さて，サブルーチンをざっと説明しておく。

**SETROM：**

DEレジスタに入っているJISコードの漢字パターンを読み出すための設定をする。この部分はリスト3-4の130〜230行に相当する部分である。

**RROM：**

漢字ROMから2バイトだけ読み出すサブルーチン。Dレジスタが左側のパターン，Eレジスタが右側のパターンの相当データを持ってリターンする。$FE98_H$番地のNOPは，その前にある二つの「DEC C」とともに，3$\mu$sのウエイトになるわけである。

**XYADDR：**

X, Y座標からグラフィックアドレスを求めるサブルーチンである。X座標は8ドットおきに限定しているので，なかなか短くなっている。

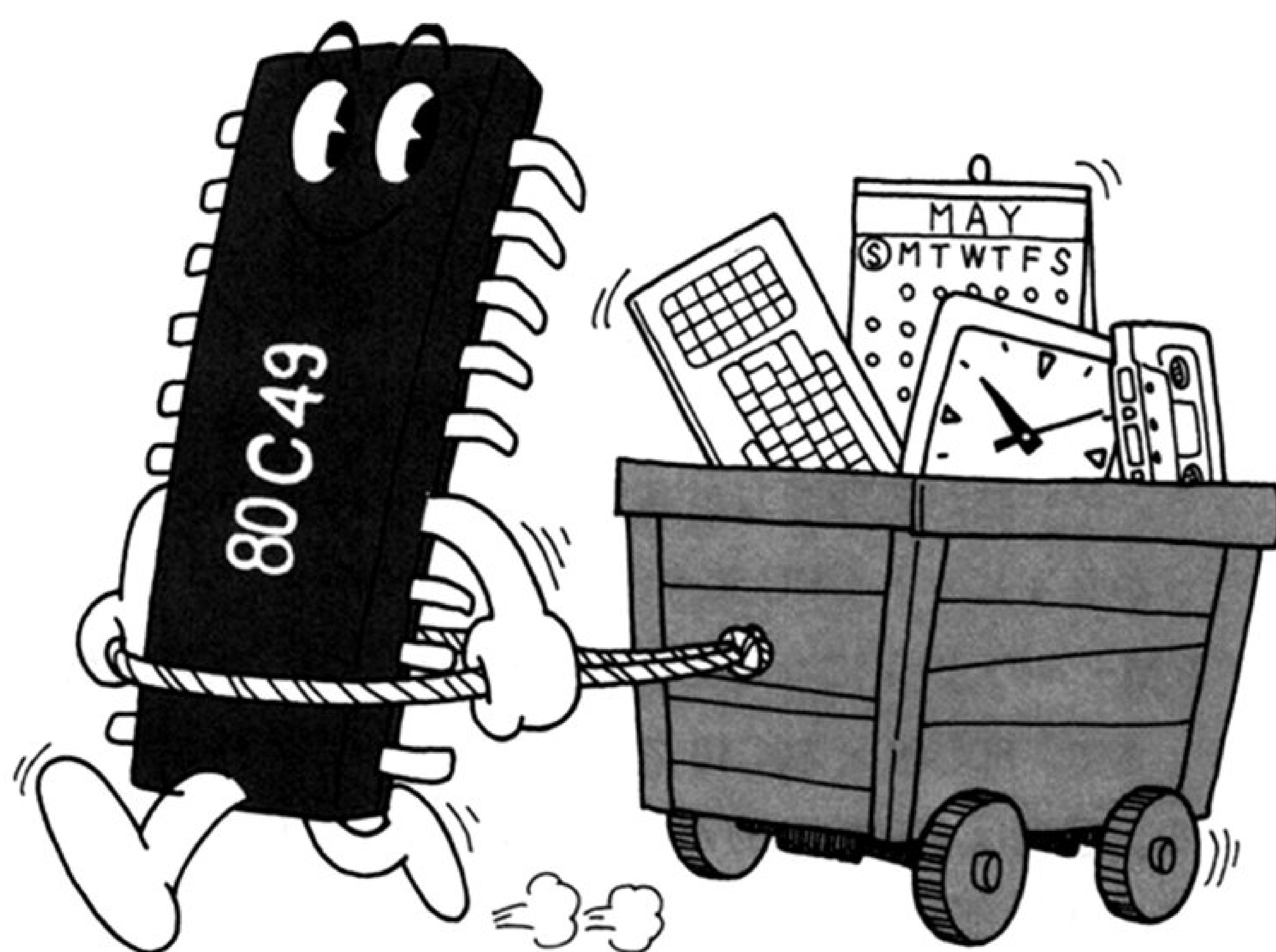
というわけであるが，最後に付け加えておくと，$FE59_H$〜$FE68_H$のあたりはグラフィックアドレスを1ライン下げるルーチンである。座標を青画面に限定しているので，このよーに短くなっている。

そいでもってリスト3-6を実行すると，JISコードを聞いてくる。で，適当なコードを入れてやるとポコッと表示するわけである。一番気になったのは速度だったのであるが，どうやら非常に速いようである。ただしそれは機械語に入ってからのことで，実際にリスト3-6を走らせてもBASICの部分がどうしようもなく遅いのであった。困ったことよ。

以上，漢字について一とおりやったわけである。しかしながら，この章でやったことは最小限のことだけである。理想を言えば，せめて単漢字変換ぐらいやっておきたいわけであるが，ソフト的なことはまったく棚に上げてしまったわけである。

# 第4章

# サブCPU




サブCPUのおかげなのである

# 第4章
# サブCPUのおかげなのである・・・

その存在が意識されることは少ないが，X1では，カセットのコントロール，キーボードからの入力，時計，カレンダーはサブCPUの縄張りになっているのである。てなわけで，この章ではその裏方さんのサブCPUをやるのである。

## 80C49なのである

まずはサブCPUそのものについて紹介しておこう。X1に使われているサブCPUは，80C48, 80C49という2種類の8ビットワンチップマイコンである。これらの石はインテル社の発表した8048，8049のC-MOS版なので，型番の間に「C」が入っている。ちなみにC-MOSの特徴の一つは消費電力が極めて少ないということである。そのことによってX1の電源がOFFになっていてもサブCPUは生きていて，キーボードからテレビをコントロールできるのである。さもなくばX1は**電気食い虫になり下がってしまう**のである。

それはさておき，80C48，80C49について基礎教養講座を開くのである。

ワンチップマイコンというものは，主に制御用に使われているLSIである。最近では家庭電気器具にマイコンを組み込むのがはやっていて，CMなどでもしょっちゅう自慢しているからおなじみであろう。

ワンチップマイコンには4ビットのものと8ビットのものがある（そのうち技術をもて余したチップメーカーが16ビットや32ビットのものを作るかもしれない)。このビット数の違いは，パソコン用のCPUの8ビット，16ビット，32ビットと同じようなものである。早い話が処理能力が違うのだ。80C48と80C49はめでたいことに8ビットである。うっかり4ビットCPUにキーボードをまかせたりしたら，ゲームをするときの**キー操作がFM-7のように**なっていたかもしれない。鶴亀鶴亀。

さて，80C48と80C49の違いであるが，これら二つのマイコンは基本的に同じなのである。序列は80C49の方が上位（高機能）である。これらのマイコンは，一つのチップの中に，CPU部分と，ROM，RAMを持っていて，基本的にプログラムや定数データはすべてチップ内のROMに入る。いわばクリーンコンピュータの完全な逆なのである。チップ内のRAMは主に変数やワークエリアとして使われる。二つのマイコンの違いはこれらのメモリの容量で，80C48のROM：1Kバイト，RAM：64バイトに対して，80C49ではそれぞれが倍になっている。ただしワンチップマイコンといっても外部にROMやRAMを付けることも可能で，そのようにすれば両方とも同じ性能にまで拡張できるそうである。詳しくは参考文献9を見ていただきたい。

## サブCPUの実態なのである

X1でのサブCPUの使われ方は，80C49が本体の中にあり，キーボード，カセットデッキ，カレンダー，時計（μPD1990というLSI）とZ80との間をとりもっている。X1のキーボードの中には80C48が入っており，押されたキーのデータを80C49がシリアル転送している。これは80C48（キーボード）から80C49（本体）への一方通行である。すなわち間に赤外線の送受信装置などを挟めば，かなり簡単にワイヤレスになるはずである。さて，turboのキーボードではちょいと違う。80C48ではなく，80C49が使われているのである。これは「Bモード」なんちゅうものがキーボードに付いたりなどしたため，80C48では処理能力が足りなくなったためだと思われる。

turboでBモードという機能が付いたことからも想像できるように，実は本体内のサブCPUの80C49も，X1とturboでは違うのである。正確に言えば，内部のROM（マスクROM）に書かれているプログラムが違うのである。さらに，turboのmodel 20/30においてはそれだけではなく，データレコーダ(CZ-8RL1)をつなぐための変更もある。8RL1にも80C49が内蔵されており，インテリジェントになっている（確認はしていないが，専用ディスプレイにも入っていると思う)。turboのmodel 20，30のサブCPUは，8RL1の80C49同士とシリアル通信を行なっているのである。これは8RL1をつなぐことができるX1F model 20でも同じことである。すなわち，本体内のサブCPUは，

1） X1 turbo model 20，30，40（II，III，Zも含む）
2） X1 turbo model 10
3） X1F，G model 20
4） その他

の4種で，それぞれ別の仕様になっているのである。もっと細かい部分や型番なども違う可能性があるが，少なくとも上記の4種は機能が違っていることは確かである。8RL1をつなげるようにした部分の変更はZ80側からはまったく見えない。すなわち，サブCPUがその部分の「違い」を完全に吸収しているのである。てなわけで，周辺機器の構成の変化にも対応できる点が，サブCPUを使うメリットでもあるのだ。

## さあて，使い方である

サブCPUを使うには基本的に2とおりの方法がある。それはサブCPUからデータを受け取るか，それともサブCPUにデータを渡すかの違いである。ただし，どちらの場合でも最初はZ80から80C49へコマンドを送ってやらなければならない。80C49はそのコマンドを見て，自分がデータをZ80に送るのか，それともZ80からデータを受け取るのかを判断するのである。なお，それぞれの場合の受け渡しをするデータ数（バイト数）も決まっているので，その数を守らないと正常な動作をしなくなってしまうので注意が必要である。というところで**表4-1**にサブCPUのコマンド表を示す。

表 4-1　サブ CPU コマンド一覧

| コマンド | 動　　　作 | データ転送方向 | データ数（バイト数） | 備　　　考 |
|---|---|---|---|---|
| $E3_H$ | ゲームキーデータ読み出し | Z80←80C49 | 3 | turboのみ。キーボードのモードがAならデータは$00_H$ |
| $E4_H$ | キー入力割り込みベクタセット | Z80→80C49 | 1 | ベクタを$00_H$にすると，割り込みは起きない |
| $E6_H$ | キーデータ読み出し | Z80←80C49 | 2 | データはファンクション部，ASCIIコードの順 |
| $E7_H$ | TVコントロール | Z80→80C49 | 1 | 特になし |
| $E8_H$ | TVコントロール読み出し | Z80←80C49 | 1 | TVに最後に送られたコードを読み出す |
| $E9_H$ | カセットデッキコントロール | Z80→80C49 | 1 | データは，CMT=～に対応 |
| $EA_H$ | カセットデッキ状態読み出し | Z80←80C49 | 1 | データは，～=CMTに対応 |
| $EB_H$ | カセットセンサー読み出し | Z80←80C49 | 1 | データは，～=CMT(△)に対応 |
| $EC_H$ | カレンダーセット | Z80→80C49 | 3 | 特になし |
| $ED_H$ | カレンダー読み出し | Z80←80C49 | 3 | 特になし |
| $EE_H$ | 時計セット | Z80→80C49 | 3 | 特になし |
| $EF_H$ | 時計読み出し | Z80←80C49 | 3 | 特になし |
| $D0_H$～$D7_H$ | タイマ(0～7)セット | Z80→80C49 | 6 | 特になし |
| $D8_H$～$DF_H$ | タイマ(0～7)読み出し | Z80←80C49 | 6 | 特になし |

では，例によってサンプルプログラムを使いながら順に解説するのであった。

### ⓪　サブ CPU と会話するにも礼儀あり

第 0 章の I/O マップを見ても分かるように，サブ CPU との会話は I/O 空間の $1900_H$番地を使って行なうのである。しかし，ただ単に $1900_H$番地にデータを IN/OUT すればよいというものではない。サブ CPU にもサブ CPU の都合というものがあり，おのずと礼儀を守らなければならないのである。早い話が $1A01_H$番地のビット 6 が 1（セットされている）ならば，サブ CPU は「今は忙しいからデータは受け取れません」といっているのである。また同じく $1A01_H$番地のビット 5 が 1（セットされている）ならば，「今はあんたに渡すデータはないです」と言っているのである。すなわち，

**$1A01_H$番地のビット 6**

　0 ならば Z80 → 80C49 へ 1 バイト

　1 ならばダメ

**$1A01_H$番地のビット 5**

　0 ならば 80C49 → Z80 へ 1 バイト

　1 ならばダメ

なのである。

そこで**リスト 4-1** を見ていただきたい私なのである。このプログラムを打ち込み間違った場合は暴走したり，キー入力が受け付けられなくなったりするので，実行前に一度 SAVE しておくことをお勧めする。さて，このプログラムはオール BASIC である。やっていることはカレンダー（日付）の読み書きなのである。問題の焦点の 80C49 とのデータのやりとりは，

400～430 行が Z80 → 80C49

リスト 4-1　オールBASIC版サブCPUアクセスルーチン

```
100 DATE$="99/02/23"
110 D$=CHR$(&HED)+"...":'GET DATE$
120 GOSUB"GET-DATA"
130 GOSUB"DISPHEX":PRINT
140 '
150 D$=CHR$(&HEC,&H85,&H93,&H18):'DATE$="85/09/18" and WED
160 GOSUB"SET-DATA"
170 END
180 '
190 LABEL"DISPHEX"
200 FORI=2TOLEN(D$)
210   D=ASC(MID$(D$,I,1)):PRINT HEX$(D),
220 NEXT:RETURN
230 '
240 LABEL"GET-DATA"
250 D=ASC(LEFT$(D$,1))
260 GOSUB"Z80->80C49"       :'SET COMMAND
270 FORI=2TOLEN(D$)
280   GOSUB"80C49->Z80"
290   MID$(D$,I,1)=CHR$(D)
300 NEXT:RETURN
310 '
320 LABEL"SET-DATA"
330 D=ASC(LEFT$(D$,1))
340 GOSUB"Z80->80C49"       :'SET COMMAND
350 FORI=2TOLEN(D$)
360   D=ASC(MID$(D$,I,1))
370   GOSUB"Z80->80C49"
380 NEXT:RETURN
390 '
400 LABEL"Z80->80C49"
410 WHILE INP(&H1A01) AND &H40:WEND:'WAIT
420 OUT&H1900,D                     :'SET 1 BYTE
430 RETURN
440 '
450 LABEL"80C49->Z80"
460 WHILE INP(&H1A01) AND &H20:WEND:'WAIT
470 D=INP(&H1900)                   :'GET 1 BYTE
480 RETURN
```

450～480 行が 80C49 → Z80

である。ラベルそのものだから別に説明する必要もないのであった。

プログラムではサブCPUとのやりとりで代表的な二つの形式を示している。まず，100～130 行がデータ入力である。これは，まずサブCPUにコマンドED$_{H}$を渡して，「カレンダーのデータをちょーだい」と言うのである。サブCPUはED$_{H}$というコマンドを受け取ると，「よっしゃよっしゃ」と言って3バイトのデータを送ってくれる。110 行の〝..."はその3バイトのデータを受け取るためのダミーである。

次に 150～160 行がカレンダーのセットである。最初のEC$_{H}$が，「これからカレンダーをセットするぞ」という意味のコマンドである。サブCPUはこのコマンドを受け取ると，またもや「よっしゃよっしゃ」と言って3バイトのデータを要求してくるのである。そこで，150 行の&H85，&H93，&H18 の3個を渡してやるのである。

このサンプルから分かるようにサブCPUとの交信には，

① **タイプ1**

1バイトのコマンドを 80C49 へ送る。

その後 80C49 が数バイトのデータを送ってくる。

② **タイプ 2**

1 バイトのコマンドを 80C49 へ送る。

その後 80C49 がさらに数バイトのデータを要求する。

の 2 とおりがあるのだ。①を 240～300 行のサブルーチン，②を 320～380 行のサブルーチンで行なっている。

補足するが，100 行では読み出しがいがあるようにデタラメな日付をカレンダーにセットしている。190～220 行のサブルーチンは D$の中身の表示である。

というところで，いきなり凶器攻撃である。実はこのプログラムはぜんぜん正しくないのだ。170 行を「GOTO 110」にしてループさせ，BREAK キーだけをハイパーオリンピック的にパシパシと叩くと，やがて BASIC が止まってしまうのである。実はこれはキー割り込みのせいなのである。そこで機械語を使った正しいサブルーチンを示さねばなるまい。**リスト 4-2** が 240～480 行と同じことをするプログラムのアセンブルリストである。以後のサンプル（リスト 4-4， 5， 6， 7， 9）では 100～170 行でこの機械語サブルーチンを使うことになる。

**リスト 4-2　サブ CPU アクセスルーチン**

```
                                 .Z80
                                 .PHASE  0E000H
                         ;
E000    C3 E006                  JP      TO49
E003    C3 E01D                  JP      FM49
                         ;
E006    FB               TO49:   EI
E007    EB                       EX      DE,HL
E008    56                       LD      D,(HL)   ;A=COMMAND
E009    23                       INC     HL       ;INC POINTER
E00A    58                       LD      E,B      ;COPY COUNTER
E00B    CD E034                  CALL    SEND1    ;SEND COMMAND
E00E    CD E046                  CALL    CANW     ;CAN SEND?
E011    F3                       DI
E012    1D                       DEC     E        ;DEC COUNTER
                         ;
E013    56               TO49LP: LD      D,(HL)   ;GET DATA
E014    CD E034                  CALL    SEND1    ;SEND 1 BYTE
E017    23                       INC     HL       ;INC POINTER
E018    1D                       DEC     E        ;DEC COUNTER
E019    20 F8                    JR      NZ,TO49LP
E01B    FB                       EI
E01C    C9                       RET
                         ;
E01D    FB               FM49:   EI
E01E    EB                       EX      DE,HL
E01F    56                       LD      D,(HL)   ;A=COMMAND
E020    23                       INC     HL       ;INC POINTER
E021    58                       LD      E,B      ;COPY COUNTER
E022    CD E034                  CALL    SEND1    ;SEND COMMAND
E025    CD E046                  CALL    CANW     ;CAN SEND?
E028    F3                       DI
E029    1D                       DEC     E        ;DEC COUNTER
                         ;
E02A    CD E03D          FM49LP: CALL    GET1     ;RECEIVE DATA
E02D    72                       LD      (HL),D   ;STORE DATA
E02E    23                       INC     HL       ;INC POINTER
E02F    1D                       DEC     E        ;DEC COUNTER
E030    20 F8                    JR      NZ,FM49LP
E032    FB                       EI
E033    C9                       RET
```

```
                          ;
E034    CD E046           SEND1:  CALL    CANW     ;SUB-CPU READY?
E037    01 1900                   LD      BC,1900H
E03A    ED 51                     OUT     (C),D    ;SEND DATA
E03C    C9                        RET
                          ;
E03D    CD E050           GET1:   CALL    CANR     ;SUB-CPU READY?
E040    01 1900                   LD      BC,1900H
E043    ED 50                     IN      D,(C)    ;GET DATA
E045    C9                        RET
                          ;
E046    01 1A01           CANW:   LD      BC,1A01H
E049    ED 78             CANWLP: IN      A,(C)
E04B    E6 40                     AND     40H
E04D    20 FA                     JR      NZ,CANWLP
E04F    C9                        RET
                          ;
E050    01 1A01           CANR:   LD      BC,1A01H
E053    ED 78             CANRLP: IN      A,(C)
E055    E6 20                     AND     20H
E057    20 FA                     JR      NZ,CANRLP
E059    C9                        RET
                          ;
                                  END
```

リスト 4-3 正しいサブ CPU の使い方

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE003
120 MEM$(&HE000,16)=HEXCHR$("C3 06 E0 C3 1D E0 FB EB 56 23 58 CD 34 E0 CD 46")
130 MEM$(&HE010,16)=HEXCHR$("E0 F3 1D 56 CD 34 E0 23 1D 20 F8 FB C9 FB EB 56")
140 MEM$(&HE020,16)=HEXCHR$("23 58 CD 34 E0 CD 46 E0 F3 1D CD 3D E0 72 23 1D")
150 MEM$(&HE030,16)=HEXCHR$("20 F8 FB C9 CD 46 E0 01 00 19 ED 51 C9 CD 50 E0")
160 MEM$(&HE040,16)=HEXCHR$("01 00 19 ED 50 C9 01 01 1A ED 78 E6 40 20 FA C9")
170 MEM$(&HE050,10)=HEXCHR$("01 01 1A ED 78 E6 20 20 FA C9")
180 '
190 DATE$="99/02/23"
200 D$=CHR$(&HED)+"...":'GET DATE$
210 DUMMY$=USR1(D$)
220 GOSUB"DISPHEX":PRINT
230 '
240 D$=CHR$(&HEC,&H85,&H93,&H18):'DATE$="85/09/18" and WED
250 DUMMY$=USR0(D$)
260 END
270 '
280 LABEL"DISPHEX"
290 FORI=2TOLEN(D$)
300  D=ASC(MID$(D$,I,1)):PRINT HEX$(D),
310 NEXT:RETURN
```

ここで，当然のことのようにリスト 4-2 を解説するのであった。使い方は**リスト 4-3** を見れば分かると思うが，念のために言うと，

E000$_H$がタイプ 1 （全部 Z80 → 80C49）

E003$_H$がタイプ 2 （最初の一つ以外は 80C49 → Z80）

のそれぞれエントリーである。DE レジスタにコマンドとデータ部分へのポインタを，B レジスタにその両方を合わせたバイト数を入れて CALL すればよい。これは BASIC で USRn (△ $) を実行したときにセットされるレジスタの内容に合わせているので，リスト 4-3 のように手軽に使えるのである。タイプ 1，タイプ 2 両方とも処理の流れは同じようなものであるから，タイプ 1 についてだけ処理を追ってみる。

まず E000$_H$でいきなり「TO49」へジャンプしてしまう。つまりそこが本当の始まりなのだ。最初に EI で割り込みを許可している。一見無意味のようだが，これには深い背景

と，私の一晩の苦しみが込められているのだ。X1では動作中ほとんどの場合において割り込みが許可（つまりEI）されているのだが，これは主にキー入力のためである。さて，前にも書いたように，そのキー入力による割り込みもサブCPUによって行なわれているのだ。このことからも分かるように，サブCPUと（キー割り込み以外のことで）交信しようとする場合は，割り込みを禁止（DI）しておかなければならないのだ。だが，そのタイミングは凶悪そのものである。すなわち，リスト4-2にもあるように，最初のコマンドを送った後，サブCPUが「またデータを送ってもいいよ」と言うまで割り込みを禁止してはいけないのだ。これはタイプ1，タイプ2の両方について言えることである。私はこれを発見するのに一晩かかってしまった。最後にはとうとう仕方なくHuMonitorを解析して謎が解けたのである。この取り決めを守らないと，リスト4-1での BREAK キーの速押しなどで動作がその筋してしまうのである。この点のほかは，RETする前にEIを実行する点以外に特に説明を必要とする部分はないと思う。しかし，縁起ものだから一応言っておくが，それぞれのルーチンは，

- ●SEND1 ：80C49からOKサインが出るまで待って1バイト送る
- ●GET1 ：80C49からOKサインが出るまで待って1バイト受け取る
- ●CANW ：80C49へデータを送れるようになるまで待つ
- ●CANR ：80C49からデータを受け取れるようになるまで待つ

となっている。BASICからUSR命令で呼び出す分にはかなり便利だろうと思う（実はBASICのIOCS中にはもっと便利なサブルーチンがあるのだが，それを使うのは軟弱であるという私の独断により，一切無視するのであった）。

では，具体的にそれぞれのコマンドについて解説を始める。

**① E$3_H$（turboのゲームキー読み取り）**

これはturboのキーボードのスイッチを，「B」にしたときに発動される秘密兵器である。簡単に言ってしまえば24個のキーを（ほとんど）同時にリアルタイムに読み出せるのである。24個のキーは固定で，テンキーとメインキーの S を中心としたQ～Cの8個，およびその周辺のキーである。具体的にどのキーかということは，**図4-1**を見ていただきたい。

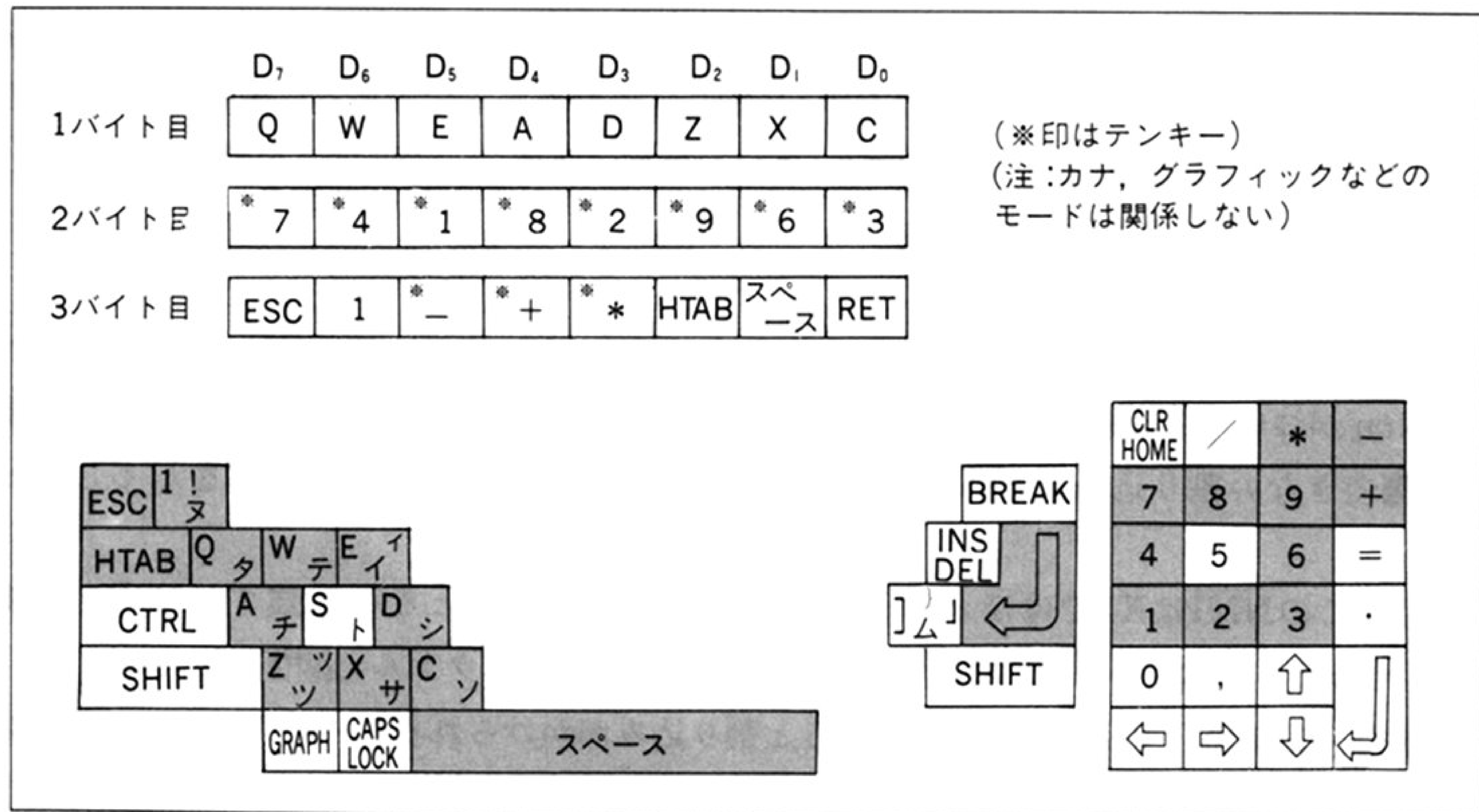
次のサンプルプログラムは**リスト4-4**である。100～170行はリスト4-3の一部分を流用したものである。230～250行は後程説明する。このプログラムではBIN$を使って，押されたキーに対応するビットを1にして表示する。ループにしてあるのでリアルタイムに変化する様子をながめていただきたい。充分味わったなら，キーボードのスイッチを「A」にしていただきたい。何を押しても全部0のままのはずである。すなわちキーボードがモードAのときは何を押しても0が返ってくるのだ。

**② E$4_H$（キー入力割り込みベクトルセット）**

出たっ！　とうとう割り込みである。この割り込みというやつはなかなかに面倒臭いので心していただきたい。

まずは一般論である。割り込みというのは，一心不乱に仕事をしている人に電話がかかってくるようなものである。その人は電話のベルを聞くと，今どの仕事のどの部分をやって

図 4-1　ゲームキー配置図




リスト 4-4　キー入力

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE003
120 MEM$(&HE000,16)=HEXCHR$("C3 06 E0 C3 1D E0 FB EB 56 23 58 CD 34 E0 CD 46")
130 MEM$(&HE010,16)=HEXCHR$("E0 F3 1D 56 CD 34 E0 23 1D 20 F8 FB C9 FB EB 56")
140 MEM$(&HE020,16)=HEXCHR$("23 58 CD 34 E0 CD 46 E0 F3 1D CD 3D E0 72 23 1D")
150 MEM$(&HE030,16)=HEXCHR$("20 F8 FB C9 CD 46 E0 01 00 19 ED 51 C9 CD 50 E0")
160 MEM$(&HE040,16)=HEXCHR$("01 00 19 ED 50 C9 01 01 1A ED 78 E6 40 20 FA C9")
170 MEM$(&HE050,10)=HEXCHR$("01 01 1A ED 78 E6 20 20 FA C9")
180 '
190 D$=CHR$(&HE3)+"...":'READ GAME KEY
200 DUMMY$=USR1(D$)
210 GOSUB"DISPBIN"
220 '
230 D$=CHR$(&HE6)+"..":'READ NORMAL KEY
240 DUMMY$=USR1(D$)
250 GOSUB"DISPHEX":PRINT
260 GOTO190
270 END
280 '
290 LABEL"DISPBIN"
300 FORI=2TOLEN(D$)
310   D=ASC(MID$(D$,I,1)):PRINT RIGHT$("0000000"+BIN$(D),8),
320 NEXT
330 RETURN
340 '
350 LABEL"DISPHEX"
360 FORI=2TOLEN(D$)
370   D=ASC(MID$(D$,I,1)):PRINT HEX$(D),
380 NEXT
390 RETURN
```

いたかを覚えておいて，電話を取るわけである。なぜ覚えておくかというと，電話が済んだ後で元の仕事に遅滞なく戻るためである。コンピュータの場合も同じようなもので，CPU は「それフェッチしてきたぞ。なになに $3E_H$ だと，よっしゃ，もう 1 バイト持ってきて A レジスタにぶち込めばよいのだな。そうれっ！　次は何だ？　おっと $C3_H$ だぜ。今日はジャンプが多いなぁ」などとつぶやきつつ，一心不乱に仕事をしているのである（たぶん）。この CPU に電話，じゃない，割り込みをかけるには，CPU からムカデ状に出ている足に電圧をかけてやればよいのである。Z80 の場合は割り込み用の足を 2 本持っている。16

番ピンと17番ピンで，おのおの$\overline{\text{INT}}$，$\overline{\text{NMI}}$と呼ばれている。上に横棒（$\overline{\text{こんなやつ}}$）が付いているのは，「0ボルトになると効きまーす」という意味なのだ。だから，このピンを0ボルト（つまりLow）にすると，CPUに割り込みがかかるのである。$\overline{\text{NMI}}$の方は例のリセットキーである(IPLじゃない方)。こいつは実に我儘で，Z80はこの割り込みに対して拒否権がないのである。すなわち，

　　Non Maskable Interrupt

なのだ。Maskには「邪魔をする」という意味がある。Maskableで「邪魔できる」。それにNonがついて「邪魔できない」。最後のInterruptは「割り込み」そのものだから，結局「邪魔できない割り込み」ということになる。う〜ん，いきなり英文読解をしてしまった。

実はこの$\overline{\text{NMI}}$はX1ではほとんど意味がないので，暇なときに勉強すれば充分である。もっと大事なのが$\overline{\text{INT}}$で，これがキー入力割り込みベクトルに関係してくる。

さて，この$\overline{\text{INT}}$の足がLowにされると割り込みがかけられるわけであるが，こっちに対してならZ80は拒否権を持っている。先程も出てきたDIである(電話の受話器を外しておくようなもの)。Z80がこの命令を実行すると，「わしゃ，$\overline{\text{INT}}$がどーなろうとも知らんもんね」を決め込むのである。これの逆がEIで，これを実行すると，「私はどんな割り込みでも受けるっ！」となる。ただし正確にはEIの次の命令の実行後である。つまり，

```
EI
RET
```

という命令が並んでいると,RET命令でサブルーチンからリターンした後で割り込みを受け付けるようになるのである。

それはさておき，こっちの割り込みには3種類があり，Z80が自由に選択できるのだ。それぞれモード0，モード1，モード2と呼ばれる（具体的には「IM　0」,「IM　1」,「IM　2」という見慣れない命令による)。このうちモード0とモード1は8080コンパチなモードである。そして実にセコイのだ。当然のごとくX1ではそんなものは使わないのである。よって，これらの二つのモードについても暇なときに勉強すればよい。というところで，$\overline{\text{INT}}$のモード2の，割り込みにやっとたどりついた私であった。

知る人ぞ知る隠れレジスタというのがZ80にある。FレジスタとIレジスタとRレジスタである。このうちIレジスタ(インタラプトレジスタ)というのがモード2の割り込みでは主役の一人である。これは8ビットのレジスタである。

モード2の割り込みが発生すると,Z80はおもむろにこのIレジスタを見るのだ。次に割り込み源（誰だっ！俺の$\overline{\text{INT}}$をLowにしたやつは？）からも1バイトのデータを受け取るのだ(これを割り込みベクトルもしくは割り込みベクタという)。次にZ80はIレジスタを上位，割り込みベクトルを下位とする2バイトの数字を作り出し，それをアドレスとみなして，その番地に何が書いてあるかを見るのである（見るだけ)。そしてZ80は**そこに書いてあるアドレスへ**サブルーチンコールを起こすのである（正確にはサブルーチンコールではないが，ほとんど同じ)。X1のBASIC（turboは違う）ではIレジスタが$00_H$で，キー入力による割り込みベクトルが$52_H$である。そこでZ80は$0052_H$番地からの2バイ

トに格納されているアドレスに飛んでいくのである。つまり，X1 の BASIC では，最終的に Z80 は，

PEEK(&H0052)＋PEEK(&H0052)＊256＝&H0346

に制御を移すのである。ここで Z80 の声を聞いてみるのである。

**Z80**：「おっ！誰でい，誰でい，おいらの $\overline{\mathrm{INT}}$ を Low にするやつは。えーと，今の割り込みモードは……ああ，モード 2 になってら。それに割り込みも許可してあらーな。よしよし，分かったから割り込みベクトルをよこしな。ふうん，$52_H$ねぇ。まあいいだろう。さてと，I レジスタの方はっと……$00_H$ねぇ。となると，ちょいと $0052_H$番地からの 2 バイトをのぞいてみるとすっか。えーとなになに，$0346_H$と書いてあるな。そんじゃちょいと一仕事とくらあ。あらよっ」

となるのであった。

それで結論である。つまりこの$E4_H$というコマンドは，そのキー入力用の割り込みベクトルを設定するものなのだ。今書いたように，X1のBASIC（CZ-8CB01，FB01のVer 1.0，2.0の四つ）ではIレジスタが$00_H$，キー入力の割り込みベクトルが$52_H$である。turbo BASIC では I レジスタが $F8_H$，キー入力の割り込みベクトルが $1A_H$である。

そこで**リスト 4-5** のサンプルプログラムを見ていただきたい。これは CB01，FB01 用で，turbo BASIC では動かない。このプログラムは，キー入力ベクトルを変えて，$0052_H$ではなく $0054_H$番地を参照するようにしている。そこには $E060_H$を書き込んであるので，結局何かキーを押すと，Z80 は $E060_H$に飛んでくることになる。$E060_H$で手ぐすね引いて待っているのは 200，210 行にある機械語プログラムである。最後にある「00 30」は変数エリアで，この I/O アドレスの所へ $65_H$（200 行の中程にある）を OUT し，変数を一つ増やして（たとえば $3000_H$→ $3001_H$）から本当のキー入力ルーチンの $3046_H$番地に飛んでいる。要するにこのプログラムを 1 回走らせて「OK」が出ると，それ以後何かキーを押すたびに画面に「e」が 1 個ずつ現れるのだ。正確には，キーを離したときにもキー割り込みが起きるので，ポンと 1 回キーを押すと「ee」と現れる。変数エリアにある $3000_H$はそのたびに増加するので，しばらく続けるとグラフィック RAM に $65_H$を書き込み始めることになり，青い点線が画面をトコトコ走るであろう。書き込みアドレスを「00 30」ではなく，最初から「00　40」にしておくという手もある。打ち間違いがあると当然キー入力ができなく

**リスト 4-5　キー割り込みにオジャマ**

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE003
120 MEM$(&HE000,16)=HEXCHR$("C3 06 E0 C3 1D E0 FB EB 56 23 58 CD 34 E0 CD 46")
130 MEM$(&HE010,16)=HEXCHR$("E0 F3 1D 56 CD 34 E0 23 1D 20 F8 FB C9 FB EB 56")
140 MEM$(&HE020,16)=HEXCHR$("23 58 CD 34 E0 CD 46 E0 F3 1D CD 3D E0 72 23 1D")
150 MEM$(&HE030,16)=HEXCHR$("20 F8 FB C9 CD 46 E0 01 00 19 ED 51 C9 CD 50 E0")
160 MEM$(&HE040,16)=HEXCHR$("01 00 19 ED 50 C9 01 01 1A ED 78 E6 40 20 FA C9")
170 MEM$(&HE050,10)=HEXCHR$("01 01 1A ED 78 E6 20 20 FA C9")
180 '
190 POKE&H54,&H60,&HE0
200 MEM$(&HE060,16)=HEXCHR$("F5 C5 ED 4B 74 E0 3E 65 ED 79 03 ED 43 74 E0 C1")
210 MEM$(&HE070,6) =HEXCHR$("F1 C3 46 03 00 30")
220 '
230 INIT
240 D$=CHR$(&HE4,&H54):'SET KEY VECT.
250 DUMMY$=USR0(D$)
```

なったりするので，リセットスイッチの助けを借りること。

### ③ $E6_H$（キーデータの読み出し）

X1では普通キー入力には割り込みを使うわけだが，別にその方法に限定されているわけではないのである。割り込みを使わずにキーを読むには，サブCPUに$E6_H$を送った後2バイトのデータを受け取ればよい。データの内容は**図4-2**に示すように，最初の1バイトがファンクション部と呼ばれるもので，早い話がBASICのINKEY$(2)と同じである。次に受け取る1バイトがASCIIコードそのものである。リスト4-4の230〜250行がサンプルである。このプログラムは割り込みと共存しており，[BREAK]キーを押すとBASICが一瞬停止するので，その状態を見ることはできない。[SHIFT]+[BREAK]も同様である。

**図4-2　キーボードからのデータの内容**

1バイト目：[ファンクション部]　2バイト目：[ASCIIコード]

ファンクション部のデータ

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| T | P | R | G | L | K | S | C |

| ビット | 得られる情報 |
|---|---|
| T | テンキー部からの入力……………… 0：有／1：無 |
| P | キー入力……………………………… 0：有／1：無 |
| R | リピート機能の有無…………………… 0：有／1：無 |
| G | [GRAPH] キーの入力………………… 0：有／1：無 |
| L | [CAPS LOCK] キーの入力………………… 0：有／1：無 |
| K | [カ ナ] キーの入力………………… 0：有／1：無 |
| S | [SHIFT] キーの入力………………… 0：有／1：無 |
| C | [CTRL] キーの入力………………… 0：有／1：無 |

さて，実はこのファンクション部はX1とturboでは少々違うのである。X1では有効なキーが押されていなければ，ファンクション部は常に$FF_H$（ビットが全部1）なのだが，turboではビット4〜0（図4-2のG〜C）の5ビットが有効なキーが押されていなくともON/OFFするようになっている。たとえばturboで[SHIFT]キーだけを押しても，それを検知できるのだ。確認する方法は簡単で，

```
10  PRINT  ASC(INKEY$(2))
20  GOTO  10
```

をRUNすればよい。キーボードのモードがA，Bいずれでも同じである。この部分は（おそらく唯一の）非コンパチ部分であろう。モードAのときはX1と同じになるようになっていれば天晴だったと思うが，実害はないようなので許してしまうのである。

ところで，当然割り込みを使った場合も示す必要があるが，ちょいと様相が異なるので最後に回すことにする。

### ④ $E7_H$，$E8_H$（TVコントロール/読み出し）

$E7_H$は大体BASICのCRT，TVPW，CHANNEL，VOLコマンドに対応する。しかしBASICの命令ではサポートされていない機能のチャンネルコール，チャンネルの順/逆送りなども実行できるのである。$E7_H$の次のデータが行なうコントロールを**図4-3**に示す。サンプルプログラムは**リスト4-6**である。これは入力されたデータを16進数とみなして$E7_H$に続けてサブCPUに送るものである。事前にCRTのスイッチを切ってから「⏎」を押すなどして楽しんでいただきたい。このプログラムでは，「.」を入力すると$E8_H$コマンドを使ってサブCPUからデータを受け取り，表示するようになっている。まかり間違っても「$E8_H$コマンドは最後に実行された$E7_H$コマンドで送られたデータを返すだけ」などとは思わな

図4-3　TVコントロールの内容

| $E7_H$＋ | | |
|---|---|---|
| | $01_H$ | 音量アップ |
| | $02_H$ | 音量ダウン |
| | $03_H$ | 音量ノーマル |
| | $04_H$ | チャンネルコール(トグル)(turboのディスプレイCZ-850Dなどのみ) |
| | $05_H$ | TV画面 |
| | $06_H$ | 音量ミュート(トグル) |
| | $07_H$ | 不明。なぜかチャンネルが4になる |
| | $08_H$ | TV/COM画面(トグル) |
| | $09_H$ | チャンネルコール(トグル)(turboのディスプレイCZ-850Dなどは受け付けない) |
| | $0A_H$ | スーパーインポーズコントラストノーマル(一気型) |
| | $0B_H$ | チャンネル順送り |
| | $0C_H$ | チャンネル逆送り |
| | $0D_H$ | パワーオフ |
| | $0E_H$ | パワーオン/オフ(トグル) |
| | $0F_H$ | スーパーインポーズコントラストダウン(一気型) |
| | $10_H$ | チャンネル1 |
| | ∫ | チャンネル2～11 |
| | $1B_H$ | チャンネル12 |
| | $1C_H$ | TV画面 |
| | $1D_H$ | COM画面 |
| | $1E_H$ | スーパーインポーズコントラストダウン |
| | $1F_H$ | スーパーインポーズコントラストノーマル |
| | $80_H$ | TVパワーオン |

注● $0A_H$と$1F_H$，$0F_H$と$1E_H$は基本的に同じ動作だが，一気型の方が素早くスーパーインポーズになる。
● $05_H$と$1C_H$の違いは不明。
● $01_H$～$1F_H$のデータに$80_H$を加えたものは，TVパワーオン後にそれぞれの動作をする。たとえば$86_H$＝$80_H$＋$06_H$は，TVパワーオン後に音声ミュートする。

リスト4-6　TVコントロール

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE003
120 MEM$(&HE000,16)=HEXCHR$("C3 06 E0 C3 1D E0 FB EB 56 23 58 CD 34 E0 CD 46")
130 MEM$(&HE010,16)=HEXCHR$("E0 F3 1D 56 CD 34 E0 23 1D 20 F8 FB C9 FB EB 56")
140 MEM$(&HE020,16)=HEXCHR$("23 58 CD 34 E0 CD 46 E0 F3 1D CD 3D E0 72 23 1D")
150 MEM$(&HE030,16)=HEXCHR$("20 F8 FB C9 CD 46 E0 01 00 19 ED 51 C9 CD 50 E0")
160 MEM$(&HE040,16)=HEXCHR$("01 00 19 ED 50 C9 01 01 1A ED 78 E6 40 20 FA C9")
170 MEM$(&HE050,10)=HEXCHR$("01 01 1A ED 78 E6 20 20 FA C9")
180 '
190 INPUT A$:IF A$="." GOTO 240
200 CODE=VAL("&H"+A$)
210 D$=CHR$(&HE7,CODE):DUMMY$=USR0(D$):BEEP
220 GOTO190
230 '
240 D$=CHR$(&HE8)+"."
250 DUMMY$=USR1(D$)
260 GOSUB"DISPHEX":PRINT
270 GOTO 190
280 '
290 LABEL"DISPHEX"
300 FORI=2TOLEN(D$)
310  D=ASC(MID$(D$,I,1)):PRINT HEX$(D),
320 NEXT:RETURN
```

いように。$E8_H$コマンドは，あくまでサブCPUからTVに最後に送られたデータを返すのである。もちろん$E7_H$で送られたものもこのうちに入るが，キーボードを使ってTVをコントロールした場合（たとえば SHIFT + 1 ）もTVに送られたデータになるのだ（この場合は$10_H$になる）。さらにTVタイマの実行時間がきてTVコントロールが起こった場合も同様である。この点を心得ておくように。

**⑤　$E9_H$，$EA_H$（カセットデッキコントロール/状態読み出し）**

$E9_H$，$EA_H$は実に単純明解なことにBASICのCMT命令と同じである。マイナーな命令だから知らない人もいるだろうから，そのよーな場合はマニュアルを読んでいただきたい。すなわち，

$E9_H$はCMT＝××

$EA_H$は○○＝CMT

に対応する。

早い話がこれだけのことなのだが，偶然にも私はX1DのBASICのマニュアルではCMTコマンドの説明が違うことを発見してしまった。すなわちX1Dでは外部テープデッキのリモートのON/OFFということになっているのだ。

**⑥　$EB_H$（カセットセンサー読み出し）**

これはBASICのCMT（△）関数に対応している。**図4-4**とBASICのマニュアルを見比べていただきたい。なお，例によってX1DのBASICマニュアルにはこれに対応するものはない。

**⑦　$EC_H$，$ED_H$（カレンダーセット/読み出し）**

BASICのDATE$，DAY$に相当するものである。データの内容は**図4-5**に示すような形式になっている。

図にあるように，1バイト（8ビット）を上下4ビットずつに分けて，それぞれの4ビットで0～9までの数値（だけ）を指定し，結局00～99（10進数）を表現する方法をBCD（2進化10進数）という。なにやら面倒臭いが，早い話4ビットなら0～15までの数値を

図 4-4　カセットセンスデータの内容

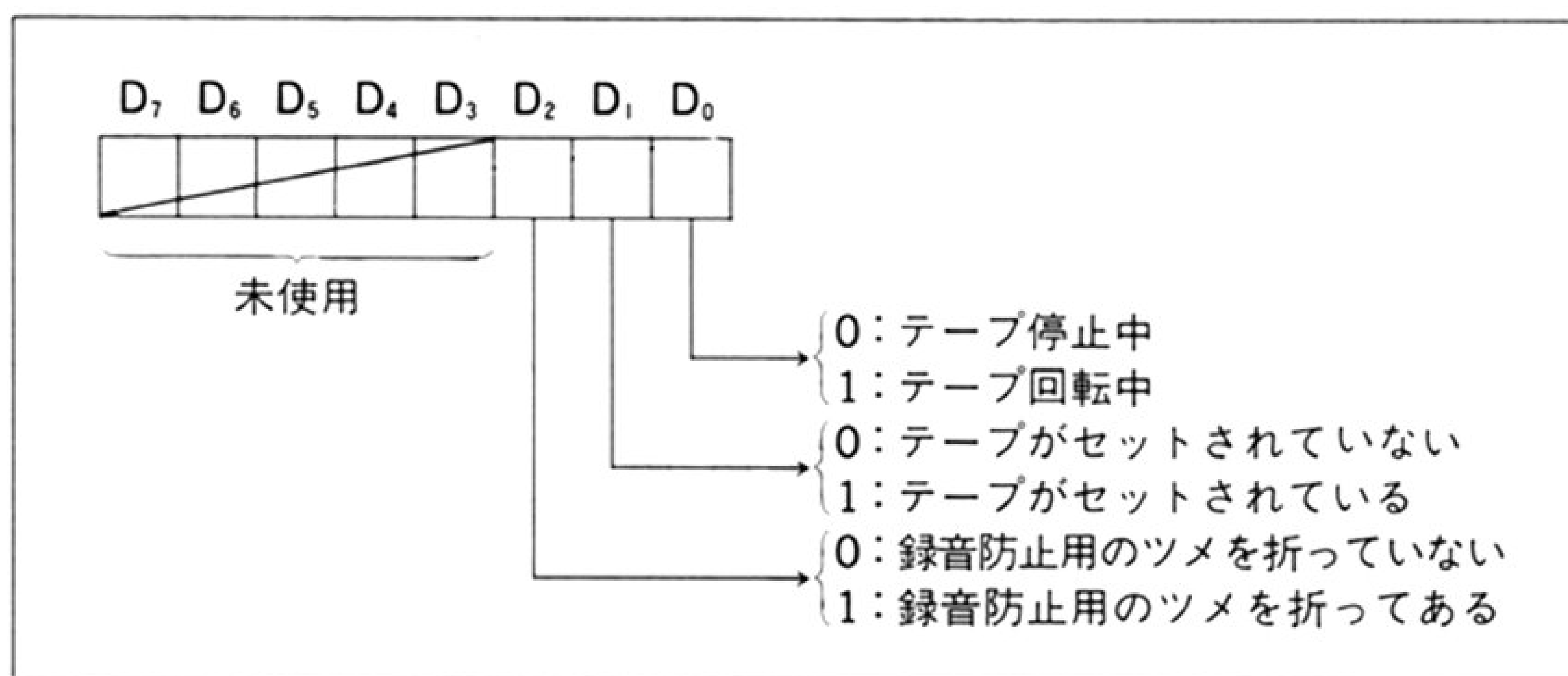
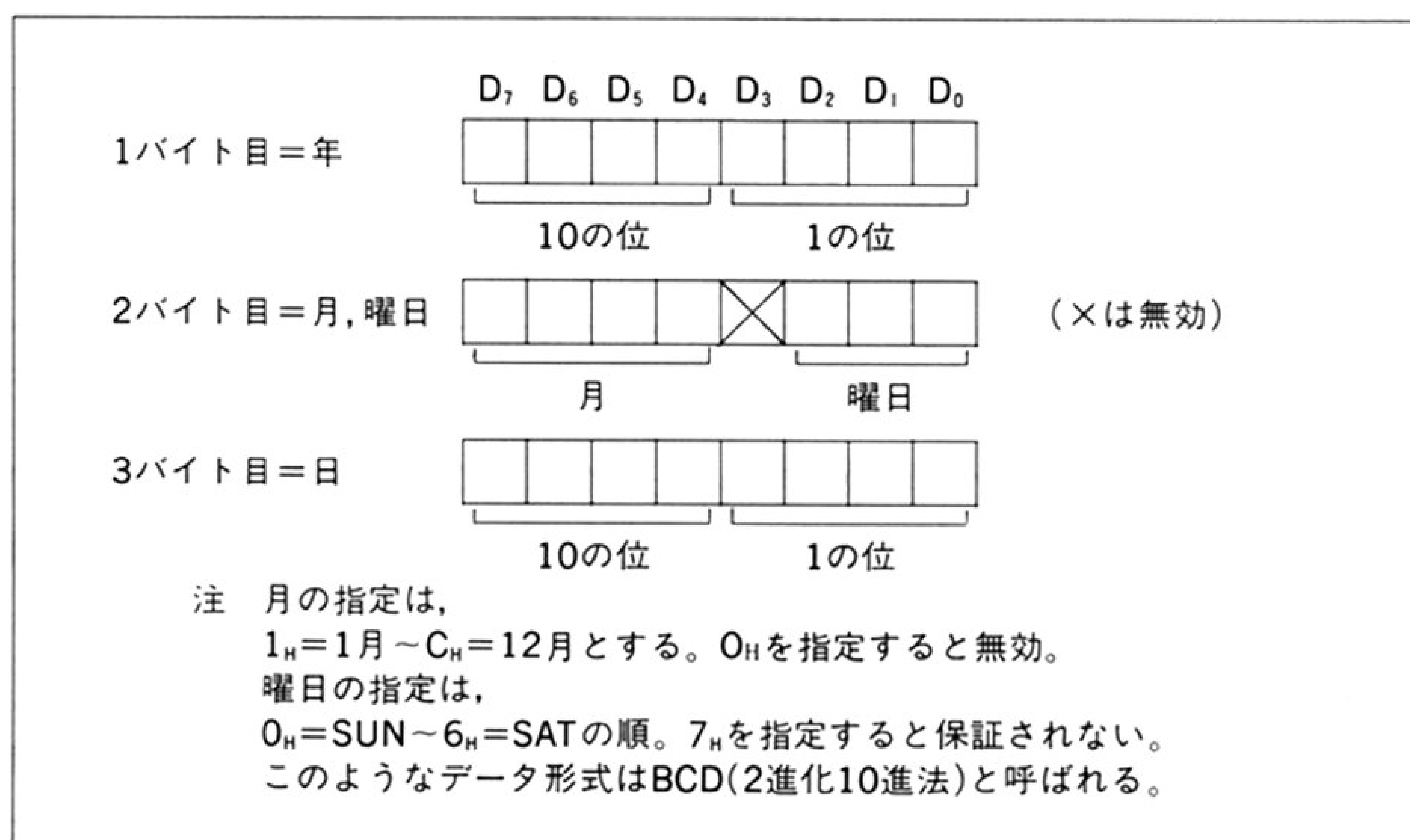



図 4-5　カレンダーデータの内容




表現できるところを 0 ～ 9 に制限して，その代わりに（ある意味で）扱いやすくしようというものである。この場合において最大の利点といえば，サンプル（リスト 4-3）にもあるように「&H」や「HEX$」を使い 16 進数のつもりで 10 進数を扱える点である。以上の説明でも理解できないなら，これでどーだ。

**「3 バイトがあるのではなく，4 ビットが 6 組あると思え」**

さて，すぐ思いつくのが，「17 月 48 日」などのめちゃくちゃな日付を設定するとどうなるかである。結果は日付がめちゃめちゃになるだけで，特に面白いことは起こらないようである。夜中の 12 時を過ぎると，存在しうる日付に訂正されてしまう。なお，大晦日を過ぎても「年」は増加しない。また，言語道断の悪習，閏年はサポートされていない。

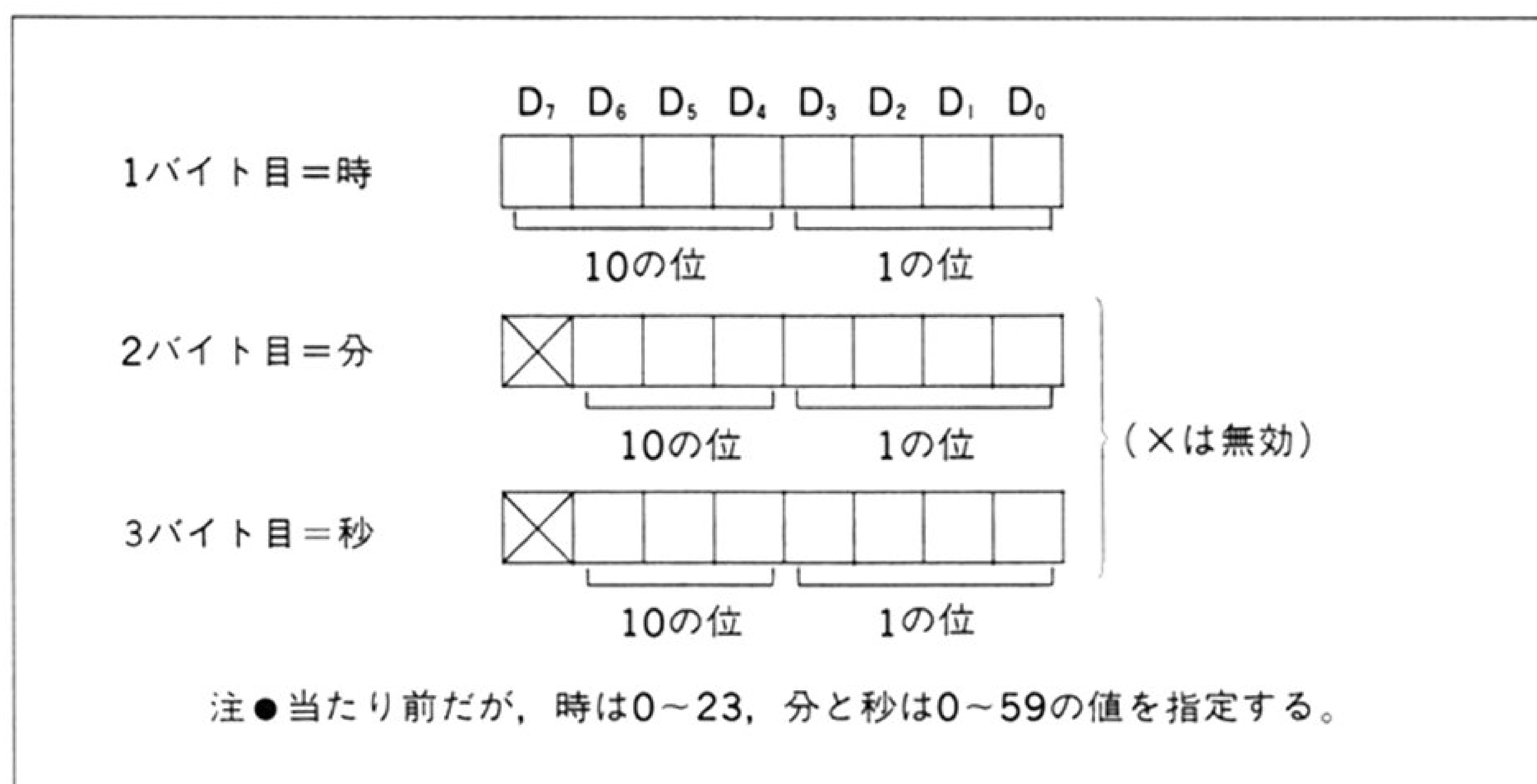
最後に一言注意。日付などは，夜中の 12 時になると，その瞬間に更新されるのではなく，数秒の遅れがあるようである。これを避けるためには，日付を読み出す直前に時計の読み出しをすればよい。そうすれば自動的に日付も時計が示すとおりに更新されるようである。

### ⑧　$EE_H$，$EF_H$（時計セット/読み出し）

BASICのTIME$に相当するものである。形式は**図 4-6** に示すとおり。

時，分，秒の数値を範囲を超えて設定しても，たちまち正常な値に訂正されてしまうようである。

図 4-6　時計データの内容




### ⑨　$D0_H$～$D7_H$，$D8_H$～$DF_H$（タイマセット/読み出し）

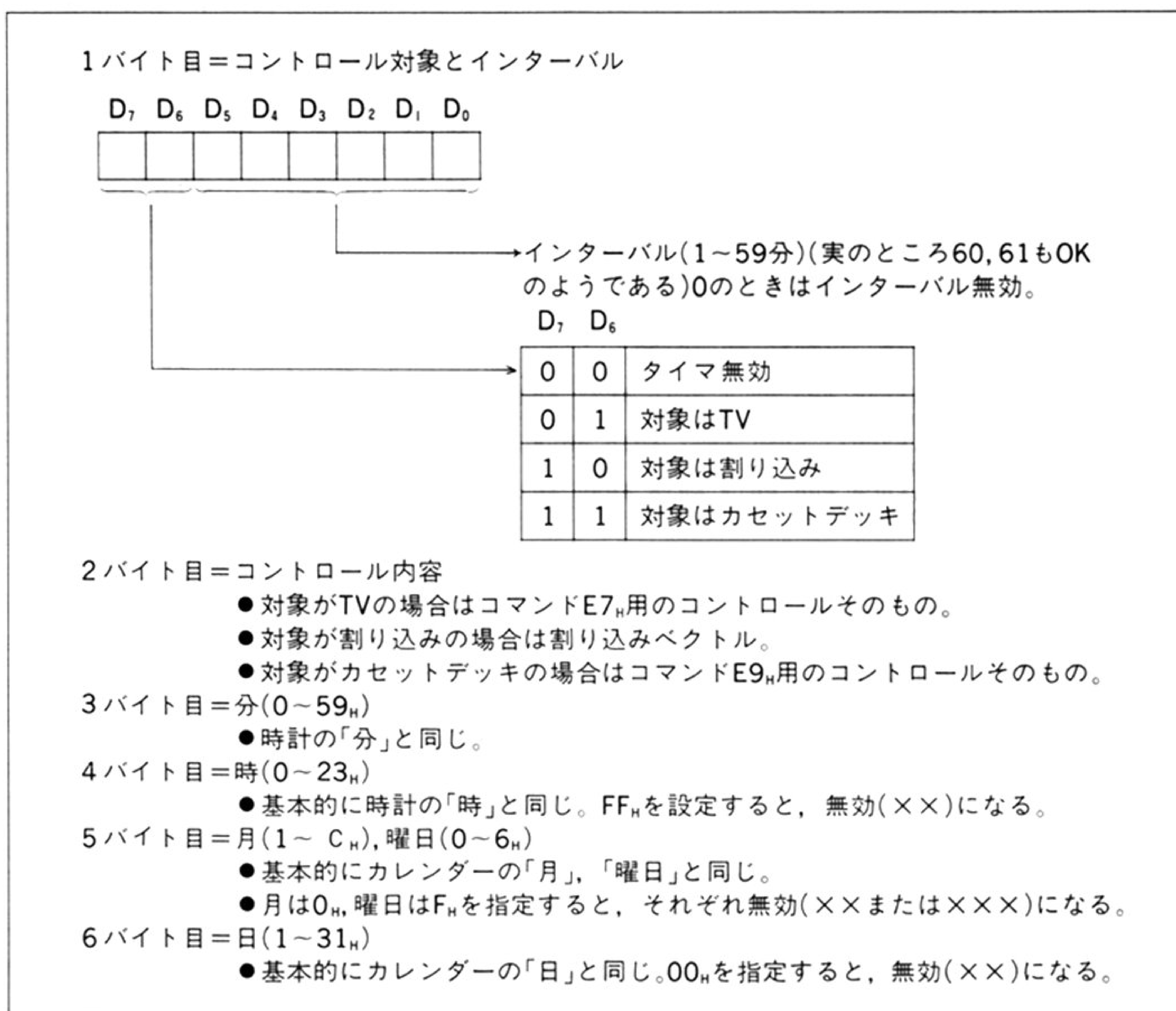
いよいよサブCPUのコマンドも最後である。しかしここで気を抜いてはいけない。なぜかというと，このTVタイマは隠れ機能の宝庫だからなのだ。そのへんは参考文献8が詳しいようである。

まず**図 4-7** を見ていただきたい。いきなり出てくる「対象は割り込み」である。そうなのである。このタイマはTVのコントロールだけではなく，割り込みやカセットデッキのコントロールもやってしまうのである。しかも，何に使ってよいやら理解に苦しむインターバル機能まである。では，おもむろに始めよう。

気付いた方もいるだろうが，実はX1のタイマは全部で8個あるのだ。ところがぎっちょん，ASKコマンドなどでタイマ設定モードに入ってもタイマは7個しか表示されない。すなわち，「タイマ0」は隠れタイマなのであった。しかもASKで設定できるのはTVコントロールのみで，さらに挙げ句の果てにはそのコントロールも$90_H$～$9B_H$および$0D_H$(パワーOFF)だけなのである。インターバルというのは，タイマに設定された時刻がきたらまず普通に（TV OFFなどを）実行し，その後はインターバルに指定された「分」ごとに同じ動作を繰り返すというものである。たとえばインターバルを1分に指定し，動作を音声ミュートにすると，この動作はトグルだから，1分ごとにTVの音が出たり出なくなったりするわけである。いたずらして友達を驚かせようとしているのはあなただけではない。

次に$D_7$，$D_6$を見ると，「10」で対象が割り込みとなる。すなわちこの場合は，タイマに設定した時刻になるとサブCPUから割り込みがかけられるのである。もちろんインターバルも有効である。このおいしそうな部分は後程ゆっくりと料理する。$D_7$，$D_6$が「11」で対象がカセットテープになる。つまり，設定された時刻になるといきなりテープが回り出すわけである。コントロール内容は$E9_H$で使えるものがすべてOKだから，突然録音状態に突

図 4-7　タイマ設定/読み出しの内容




入するなどという凶悪な事態も存在し得るのである。おーこわ。

そのよーな状態であるから，もし万が一，暴走によって

**タイマが異常にセットされた**

ならば，ゲームがいきなり止まってしまう，などということがあり得る。そのよーな症状がある場合には次のようなおはらいをしていただきたい。

1）本体裏面にあるメイン電源スイッチを OFF にする。

2）ポンポンと 2 回柏手を打つ。

3）メイン電源スイッチを ON にする。

以上でサブ CPU に設定されたタイマはすべてクリアされる。

先程も書いたように，タイマ 0 は ASK コマンドによるタイマ設定モードにおいても表示されない隠れタイマであるから，そのよーな意味も含めて 1 か月に一度ぐらいは以上のような **おはらいを励行してもよい** のではないかと思う私であった（タイマ LED がついていないのにやるのもナンだけど）。

というところで，サンプルプログラムは**リスト 4-7** である。やっていることは，まず 230 行で日付と時刻を 9 月 18 日の 16 時 10 分 55 秒にしている。タイマ 1 は，16 時 11 分に設定されており，動作は 08$_H$の TV/COM 画面トグルである。タイマ 2 は割り込みで,「&H80＋

**リスト 4-7　タイマの使い方３態**

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE003
120 MEM$(&HE000,16)=HEXCHR$("C3 06 E0 C3 1D E0 FB EB 56 23 58 CD 34 E0 CD 46")
130 MEM$(&HE010,16)=HEXCHR$("E0 F3 1D 56 CD 34 E0 23 1D 20 F8 FB C9 FB EB 56")
140 MEM$(&HE020,16)=HEXCHR$("23 58 CD 34 E0 CD 46 E0 F3 1D CD 3D E0 72 23 1D")
150 MEM$(&HE030,16)=HEXCHR$("20 F8 FB C9 CD 46 E0 01 00 19 ED 51 C9 CD 50 E0")
160 MEM$(&HE040,16)=HEXCHR$("01 00 19 ED 50 C9 01 01 1A ED 78 E6 40 20 FA C9")
170 MEM$(&HE050,10)=HEXCHR$("01 01 1A ED 78 E6 20 20 FA C9")
180 '
190 POKE &H54,&H60,&HE0:'SET INT. TABLE
200 MEM$(&HE060,10)=HEXCHR$("F5 3E 01 32 70 E0 F1 FB ED 4D")
210 POKE &HE070,0:'INIT INT. FLAG
220 CLS4:WIDTH80
230 DATE$="85/09/18":TIME$="16:10:55"
240 '
250 D$=CHR$(&HD1,&H40,&H8,&H11,&H16,&H9F,&H18):'COM. SCR. ON 16:11:00
260 DUMMY$=USR0(D$)
270 D$=CHR$(&HD2,&H80+1,&H54,&H12,&H16,&H9F,&H18):'INTERRUPT ON 16:12:00
280 DUMMY$=USR0(D$)
290 D$=CHR$(&HD3,&HC0,&H0,&H13,&H16,&H9F,&H18):'EJECT ON 16:13:00
300 DUMMY$=USR0(D$)
310 '
320 D$=CHR$(&HD8+4)+"......":'READ TIMER
330 DUMMY$=USR1(D$)
340 GOSUB"DISPHEX":PRINT
350 '
360 ADR=&HE070:KAISU=0
370 LOCATE 0,1:PRINT TIME$
380 IF PEEK(ADR) THEN POKE ADR,0:KAISU=KAISU+1:'CHECK INT.
390 PRINT KAISU:'ﾜﾘｺﾐ ﾉ ｶｲｽｳ
400 GOTO 370
410 END
420 '
430 LABEL"DISPHEX"
440 FORI=2TOLEN(D$)
450  D=ASC(MID$(D$,I,1)):PRINT HEX$(D),
460 NEXT:RETURN
```

1」となっていることから分かるように，1分のインターバルを指定している。すなわち，16時12分以降1分ごとに割り込みがかけられるわけである。タイマ3はカセットの制御で，16時13分にEJECTが行なわれる。TVコントロールとカセットの制御は見れば分かるだろうから，タイマによる割り込みを少し説明する。割り込みベクトルは$54_H$である。すなわち割り込みが発生したら$0054_H$番地に書かれているアドレスへ飛ぶ。そこで190行では，その飛び先である$E060_H$を書き込んである。実際に割り込みを処理するルーチンは200行にある。実に簡単なプログラムで，

```
PUSH  AF
LD    A, 1
LD    (0E070H), A
POP   AF
EI
RETI
```

となっている。これにより割り込みがかけられると$E070_H$番地に1がセットされる。そこで割り込みが発生したかどうかをチエックするBASICプログラムが360〜400行の無限ループである。ずっと$E070_H$番地の内容をPEEKを使って見張っていて，1がセットされたら検知するようにしている。

さて，正しいその筋の読者はここであるアイデアを思いつかなければならない。つまり，「16時12分に割り込みがかかったなら，その割り込み処理ルーチンの中で，時計を16

時 11 分 59 秒に設定すれば，その 1 秒後にまた割り込みがかかるに違いない。時計は狂ってしまうが，分単位ではなく秒単位で割り込みをかけることができる！」。

ところが残念ながら，そうはいかない。時計をしょっちゅう読み出し続けていれば（たとえば，PRINT TIME$の繰り返し）2 秒ごとの割り込みなども可能なようだが，さもなくば 1 分未満の間隔での割り込みは（まともには）無理のようである。

さらに複数のタイマが同時に割り込みを起こす場合もあり得るが，そのような状況に対してはなんらかの優先順位に基づいて，ちゃんとそれぞれ割り込みを起こすようである。

最後になったが，**リスト 4-8，4-9** である。リスト 4-8 はキー割り込みの処理ルーチンのアセンブルリストである。見て分かるように，実はこの場合はキーデータを読み出すのに際して，サブ CPU にコマンド $E6_H$を送っても送らなくてもよいのだ。また，turbo では RETI する前に，サブ CPU に $E3_H$を送ってやればゲームキーも読める。リスト 4-9 がこのルーチンを使ったサンプルである。割り込みベクトル $54_H$を使い，$E060_H$～のルーチンでキー入力を処理している。[ESC] を押すとベクトルを $52_H$にして終わる。BASIC では，バッファを持っていて，キー入力をためておくことができるのだが，このルーチンではそんなことはせずに，$E080_H$番地からの 2 バイトにファンクション部と ASCII 部のデータを書き込んでいるだけである。

てなわけであった。いざ自力でオールマシン語のゲームでも作ろうとすると，サブ CPU は案外，**避けて通れぬドラゴン**だったりする。この章を噛みしめて，力一杯その筋していただきたい。

**リスト 4-8　キー割り込み処理ルーチン**

```
                                     .Z80
                                     .PHASE  0E060H
                            ;
E034                        SEND1    EQU     0E034H
E03D                        GET1     EQU     0E03DH
E080                        KEYSTR   EQU     0E080H
                            ;
E060    F5                  INTK:    PUSH    AF
E061    C5                           PUSH    BC
E062    D5                           PUSH    DE
E063    E5                           PUSH    HL
                            ;
                            ;        LD      A,0E6H   ;D=COMMAND
                            ;        CALL    SEND1    ;SEND COMMAND
                            ;
E064    1E 02                        LD      E,2
E066    21 E080                      LD      HL,KEYSTR
E069    CD E03D             DATA2:   CALL    GET1     ;RECEIVE DATA
E06C    72                           LD      (HL),D   ;STORE DATA
E06D    23                           INC     HL       ;INC POINTER
E06E    1D                           DEC     E        ;DEC COUNTER
E06F    20 F8                        JR      NZ,DATA2
                            ;
E071    E1                           POP     HL
E072    D1                           POP     DE
E073    C1                           POP     BC
E074    F1                           POP     AF
E075    FB                           EI
E076    ED 4D                        RETI
                            ;
                                     END
```

リスト 4-9　BASIC から「リスト 4-8」を使う

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE003
120 MEM$(&HE000,16)=HEXCHR$("C3 06 E0 C3 1D E0 FB EB 56 23 58 CD 34 E0 CD 46")
130 MEM$(&HE010,16)=HEXCHR$("E0 F3 1D 56 CD 34 E0 23 1D 20 F8 FB C9 FB EB 56")
140 MEM$(&HE020,16)=HEXCHR$("23 58 CD 34 E0 CD 46 E0 F3 1D CD 3D E0 72 23 1D")
150 MEM$(&HE030,16)=HEXCHR$("20 F8 FB C9 CD 46 E0 01 00 19 ED 51 C9 CD 50 E0")
160 MEM$(&HE040,16)=HEXCHR$("01 00 19 ED 50 C9 01 01 1A ED 78 E6 40 20 FA C9")
170 MEM$(&HE050,10)=HEXCHR$("01 01 1A ED 78 E6 20 20 FA C9")
180 '
190 POKE&H54,&H60,&HE0
200 MEM$(&HE060,16)=HEXCHR$("F5 C5 D5 E5 1E 02 21 80 E0 CD 3D E0 72 23 1D 20")
210 MEM$(&HE070,8) =HEXCHR$("F8 E1 D1 C1 F1 FB ED 4D")
220 MEM$(&HE080,2) =CHR$(0,0):'CLEAR WORK AREA
230 '
240 INIT
250 D$=CHR$(&HE4,&H54):'SET KEY VECT.
260 DUMMY$=USR0(D$)
270 '
280 PRINT RIGHT$("0"+HEX$(PEEK(&HE080)),2),:'PRINT FUNC. PART
290 CODE=PEEK(&HE081)
300 PRINT RIGHT$("0"+HEX$(CODE),2),:'PRINT ASCII PART
310 PRINT#0 CHR$(CODE)              :'DISP CHARACTER
320 IF CODE=&H1B THEN D$=CHR$(&HE4,&H52):DUMMY$=USR0(D$):END:'IF "ESC" END
330 GOTO 280
```

# 第5章

# CTC




CTCは律儀なのである

# 第5章

# CTCは律儀なのである・・・・・・・・・

この章ではCTCをやるのである。CTCとはCounter Timer Circuitの略である。

まず最初に断っておくが，残念なことにNon turboのX1にはCTC，およびSIO，DMAが付いていない。しかし，オプションのCZ-8BM2を付ければ，そのボードにはCTCとSIOが載っているという次第である（CTCはCZ-8BM2以外でも，立体視ボード，FM音源ボードに載っている）。このCZ-8BM2は，RS-232Cが1チャンネルとマウスインターフェイスが付いた，なかなかのボードである。CTCとSIOはそのボードの中で主役を演じているわけであるが，はっきり言って，このボードはturboのCTC，SIO周りの機能とほとんど同じなのである。唯一の違いは，I/Oアドレスである。第0章でも示したが，turboでは，

SIO $=1F90_H \sim 1F93_H$

CTC $=1FA0_H \sim 1FA3_H$

だったのが，CZ-8BM2では，

SIO $=1F98_H \sim 1F9B_H$

CTC $=1FA8_H \sim 1FAB_H$

となっただけである。ただしこれは工場出荷時の設定であるから，ショートピンを差し替えればturboとまったく同じアドレスにすることも可能である。念のために**表5-1，5-2**に示しておく。

表5-1 SIOアドレス

| アドレス | 内　　容 | |
|---|---|---|
| $1F90_H$($1F98_H$) | チャンネルA　データポート | IN/OUT |
| $1F91_H$($1F99_H$) | チャンネルA　制御語 | IN/OUT |
| $1F92_H$($1F9A_H$) | チャンネルB　データポート | IN/OUT |
| $1F93_H$($1F9B_H$) | チャンネルB　制御語 | IN/OUT |

表5-2 CTCアドレス

| アドレス | 内　　容 | |
|---|---|---|
| $1FA0_H$($1FA8_H$) | チャンネル0 | IN/OUT |
| $1FA1_H$($1FA9_H$) | チャンネル1(SIOチャンネルA用クロック) | IN/OUT |
| $1FA2_H$($1FAA_H$) | チャンネル2(SIOチャンネルB用クロック) | IN/OUT |
| $1FA3_H$($1FAB_H$) | チャンネル3 | IN/OUT |

そこでどのよーな方針でやるかであるが，まずはCTCの割り込みを使って**「タイムシェアリングもどき」**をやってしまうのである。本当はそんなたいそーなことではなくて，音楽を鳴らすだけなのだが，基本は同じなので大風呂敷を広げて**景気をつけてしまうのである。**

なお，この章では参考文献4があると便利である。

## CTCの概略である

Z80 CTCには0番から3番まで，四つのチャンネルがある。ここでいうチャンネルとは，言ってみればそれぞれが独立した時計/カウンタのようなものである(本当は完全に独立しているわけではないのだが)。ざっと説明すると，X1では，これら四つのチャンネルは二つのグループに分かれる。すなわちチャンネル0，3とチャンネル1，2である。

X1ではチャンネル0，3は主にタイマとカウンタに使われている。早い話がZ80に対して一定時間ごとに割り込みをかけてくれるのである。一定時間ごとに割り込みをかけてくれるとどのよーなメリットがあるかというと，たとえばこれからやるような，音楽を鳴らす場合である。つまり，Z80が別のことに熱中をしていても，CTCが「あなた，そろそろPSGさんに次のデータを渡す時間ですよ」などと教えてくれるのである。つまりは秘書のようなものなのだ。秘書がいなければ，自分でしょっちゅう時計を見ていなければいけないし，うっかりすれば決められた時間を過ごしてしまうかもしれないのである。チャンネル0，3はそのように使われているのである。

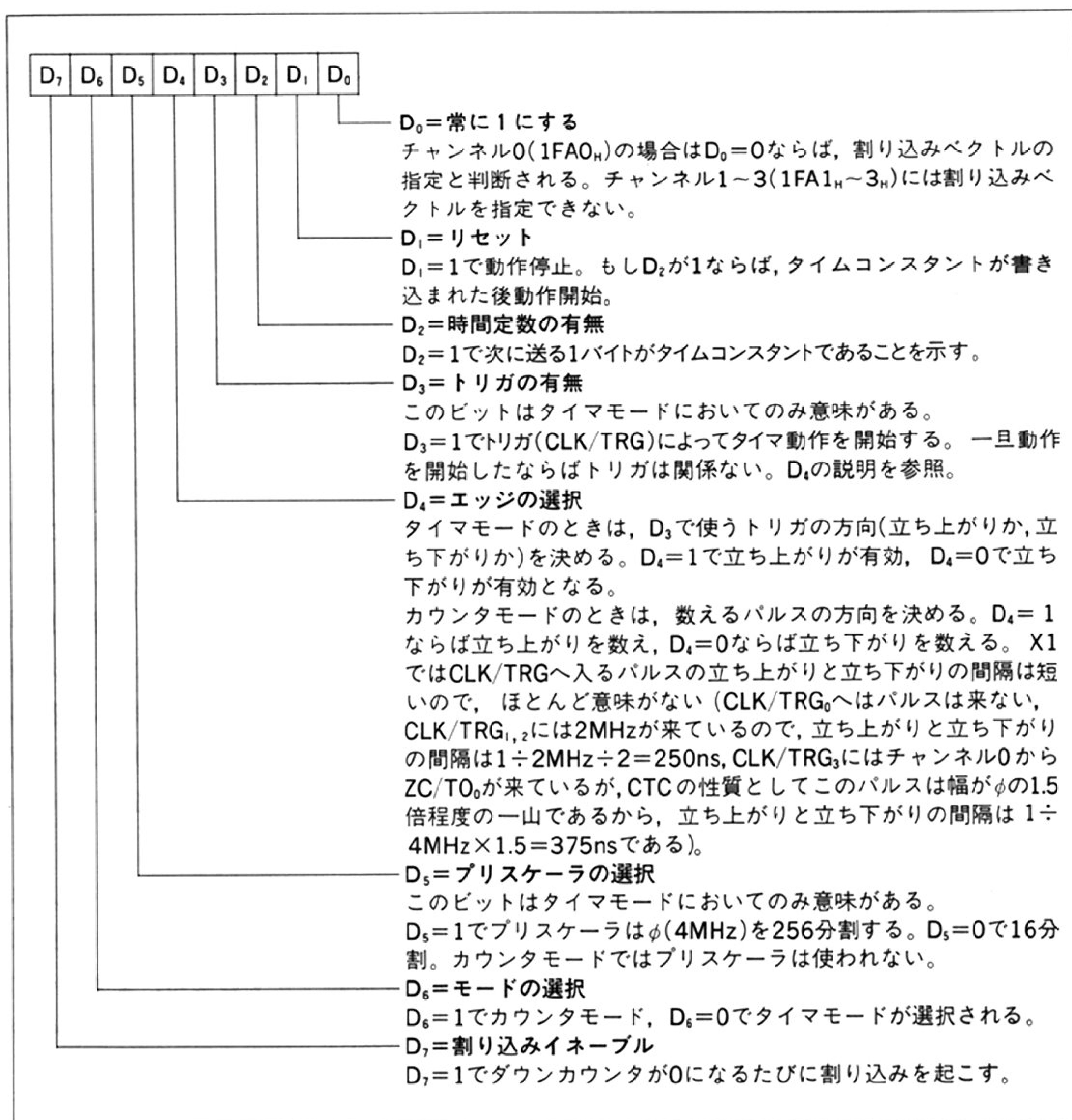
それに対して，X1ではチャンネル1，2はSIOにつながれている。具体的に何をしているかというと，RS-232Cは300 bpsとか，1200 bpsとかの転送速度があるが，そのための基準クロックを作っているのだ。すなわち，X1でSIOを使おうとするなら，まずはCTCが第一関門になっているのである。そこいらへんのことは，次の第6章でつまびらかになるであろう。

## CTCなのである

CTCのコマンドはそれほど複雑ではない。**図5-1**にあるようにコマンドは1バイトである。これを，四つのチャンネルに別々に送るわけである。指定した場合は，コマンドの直後に「タイムコンスタント」と呼ばれる1バイトを送ることもあるし，チャンネル0に対しては「割り込みベクトル」を送ることもあるが，結局はそれだけである。

まず，CTCのやることを，ごくごく簡単に説明しておく。最初に，**図5-2**を見ていただきたい。これはCTCの接続図である。「$CLK/TRG_n$」とか「$ZC/TO_n$」などがあるが，これはCTCの端子名である。「$CLK/TRG_n$」とは「外部クロック/タイマ・トリガ」ということで，要するに各チャンネルごとにある入力である。「$ZC/TO_n$」は「ゼロカウント/タイム・アウト」で，こちらはそれぞれのチャンネルの出力に対応する（ただしチャンネル3用の$ZC/TO_3$はピンの数の都合によって，省かれている)。

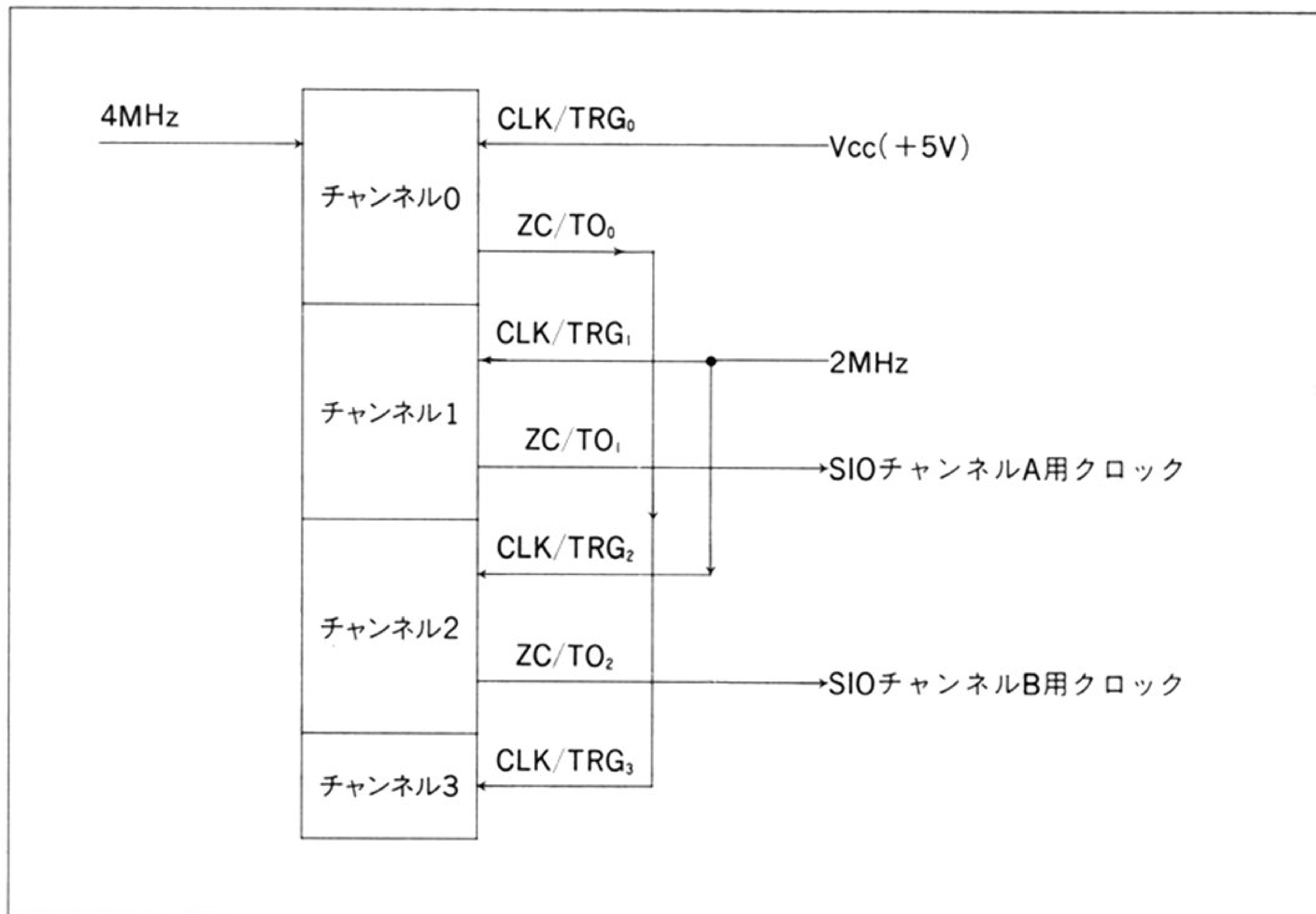
図 5-1　CTC のチャンネルコントロールワード




CTC のやることは，パルスを数えることである。数えられるパルスは図 5-2 の「CLK/TRG$_n$」もしくは，システムクロックの $\phi$（X1 では 4MHz）である。CTC の各チャンネルには「ダウンカウンタ」というものがあり，最初は「ダウンカウンタ＝タイムコンスタント」とするのである（タイムコンスタントは 0～255 を指定してやる)。各チャンネルは，それぞれがそれぞれのパルスを数えるのだが，何個かのパルスが来るたびに（1個，16個，256個の三つの場合がある）ダウンカウンタを一つずつ減らしていくのである。そして，ダウンカウンタが 0 になると（最初が 0 だったならば，256回減らすことになる)，「ZC/TO$_n$」から一つパルスを出すのである。つまり，パルスの数を 1/n にするのである。もちろんそれだけではどーしょーもないので，「ZC/TO$_n$」からパルスを出すのといっしょに，CPU に対して割り込みをかけたりもするのである。ま，これが CTC のやっていることである。

コマンドは各ビットごとに意味を持っているので，$D_7$から順に説明していくのである。

図 5-2　X1/turbo における CTC の各チャンネルの構成




$D_7$はダウンカウンタが0になったときに，割り込みを起こさせるかどうかのフラグである。

$D_6$はモードの設定である。それぞれのチャンネルに「カウンタモード」と「タイマモード」のどちらかを設定するわけであるが，「カウンタとタイマはどこがどー違うんでぃ」と，むっとする人が多いであろう。私もむっとしている。分かりやすく言うと，

「タイマモードでは，CTC に供給されているクロック（X1 では $\phi$=4MHz）を数える」

「カウンタモードでは，CLK/$TRG_n$という端子に来るパルスを数える」

というだけのことである。

$D_5$ は $D_6$ でタイマモードを選択したときにのみ意味を持つ。これは，プリスケーラの指定で，例のダウンカウンタの値を勝手に 16 倍（$D_5$＝0），もしくは 256 倍（$D_5$＝1）するものだと思えばよい。なぜそうなっているかというと，タイマモードのときの入力の $\phi$ は，一般的に 4MHz などという高い周波数なので，0～255 までしかないダウンカウンタで数えても，数え終わるのにもっとも長くて 250 ns×256＝64 $\mu$s しかなく，あまり使い道がないから，という Zilog の親心だと考えられる（本当の親心ならダウンカウンタを 2 バイトで指定できるようにすりゃいいのに。ぶつぶつ）。まあ，ここはちょっとセコイ部分である。

$D_4$ はエッジの選択である。ここは（X1 では）重要ではないので，図 5-1 を見ておくれ。

$D_3$ も X1 ではあまり重要ではない。

$D_2$ は次にタイムコンスタントを送るかどうかのフラグである。第 8 章でやる Z80 DMA

の用語で言えば，ポインタビットというやつである。

$D_1$ はリセットであるが，ちょっと注意が必要である。つまり，CTC はリセットをかけられなかったなら，そのときやっている動作を終えてから，新しいコマンドに基づく動作を開始するのである。よって，リセットは大いに使うべきである。

$D_0$ は，コマンドの場合には，常に 1 にしておく。さもなくば，割り込みベクトルを指定されたものと解釈される。

## 割り込みベクトル

CTC では，割り込みベクトルは，チャンネル 0 に対してのみ指定できることになっている(つまり，割り込みベクトルを OUT できるのは $1FA0_H$番地にのみ)。それじゃ，他のチャンネルは割り込みをかけられないのかと，一瞬思ってしまうだろうがご安心。チャンネル 0 に対して指定すると，他の 3 チャンネルにも自動的に割り込みベクトルが割り当てられるのである。すなわち，連続した四つのベクトル（8 バイト）がまとめて指定されるのである。ただし，その 8 バイトには少々制限がある。それはどーゆーことかというと，チャンネル 0 のベクトルは，

&B?????000

のよーに，下 3 ビットが 0 になっていなければならないのだ。このとき各チャンネルのベクトルはそれぞれ，

1 →&B?????010

2 →&B?????100

3 →&B?????110

となる。

だから単純に，「おっ，ここに 8 バイト分の空きがある。よしよし，CTC の割り込みテーブルに使ってしまおう」などということは許されないのだ。つまり，その 8 バイトの先頭が，

$\times\times\times 8_H$ か$\times\times\times 0_H$ （下 3 ビットが 0）

でなければいけないのだ。この点，注意が必要である。

## CTCの使い方である

さて，あーだこーだと説明してきたが，X1 の場合はハード的な CTC の使い方が単純なので，悩むことはぜんぜんないのである。つまり，本来の CTC は $D_3=1$ とすることで，「信号（トリガ）が立ち上がってから，一定時間後に割り込みをかける」などという機能があるのだが，そんなことは X1 では使えないのである(使う必要もないだろう)。X1 での CTC の役割は極めて簡単明瞭で，次の三つに要約できる。

① 図 5-2 に示したように， X1 ではチャンネル 0 用の $CLK/TRG_0$ は Vcc（+5V）につながっている。すなわち，パルスがぜんぜん来ないので，チャンネル 0 はタイマモード

で使うしかない。その場合，チャンネル 0 は，4 μs〜16.384 ms ごとに割り込みをかけてくれる。この時間が短すぎるか，ちょうどピタリの間隔が得られないのならば，$ZC/TO_0$ が $CLK/TRG_3$ に接続されていることにより，チャンネル 3 をカウンタモードで使ってチャンネル 0 と組み合わせると，最長 4.194304 秒の間隔を得られる。また，チャンネル 0，3 を別々に動かすことも可能である。

② チャンネル 1，2 はそれぞれ SIO のチャンネル A，B のクロックにつながっているので，それなりに使ってやる。

ただし，気が向いたならばチャンネル 0，3 と同じように，割り込みを起こさせることもできる（$CLK/TRG_{1,2}$ には 2MHz が来ているのでチャンネル 0 よりも割り込み間隔を短くできる)。

③ OUT 命令ではなく，IN 命令で CTC にアクセスすると，そのときのダウンカウンタの値を得られる。ただし，これはあまり使い道があるとは考えられない。

そういうわけで実際に使ってみるのである。最初は，③に示したダウンカウンタを読み出すのをやってみる。これなら割り込みを使う必要がないので入門としてはおあつらえむきである。サンプルプログラムは**リスト 5-1** である。

**リスト 5-1 ダウンカウンタを表示する**

```
100 OUT &H1FA0,3:'ﾁｬﾝﾈﾙ0 RESET
110 OUT &H1FA1,3:'ﾁｬﾝﾈﾙ1 RESET
120 OUT &H1FA2,3:'ﾁｬﾝﾈﾙ2 RESET
130 OUT &H1FA3,3:'ﾁｬﾝﾈﾙ3 RESET
140 '
150 OUT &H1FA0, &B100111:OUT&H1FA0,200:'ﾁｬﾝﾈﾙ0 SET
160 OUT &H1FA3,&B1000111:OUT&H1FA3,200:'ﾁｬﾝﾈﾙ3 SET
170 '
180 PRINTINP(&H1FA0),INP(&H1FA3):GOTO 180
```

100〜130 行では縁起ものなので，各チャンネルに 3 を OUT してリセットしている。次に 150 行でチャンネル 0 に OUT しているのは，&B00100111 である。

順に見ていくと，

- 割り込みなし
- タイマモード
- プリスケーラは 256 分割
- 無視してよい
- 無視してよい
- 次にタイムコンスタントあり
- リセット（$D_2$ = 1 だからタイムコンスタントを書き込まれた後で動作開始)
- $D_0$ だから 1

となっている。だから次に OUT している「200」はタイムコンスタントである。160 行ではチャンネル 3 に&B01000111 を OUT している。これも順に見てゆくと，

- 割り込みなし
- カウンタモード

●プリスケーラは関係なし
●無視してよい
●関係なし
●次にタイムコンスタントあり
●リセット（$D_2$ = 1だからタイムコンスタントを書き込まれた後で動作開始）
● $D_0$ だから1
である。150行と同じく，次にOUTしている「200」はタイムコンスタントである。
　このようにCTCを設定してやると，どうなるかというと，

1）　チャンネル0はタイマモードであるから，システムクロックの $\phi$（4MHz，周期は250 ns）を数える。プリスケーラが256，タイムコンスタントが200であるから，$ZC/TO_0$ からパルスを出すのは，

　250(ns)×256×200＝12.8(ms)

となる。要するに，0.00000025秒周期のパルスをもとにして，0.0128秒周期のパルスを作っているのである。

2）　チャンネル3はカウンタモードだから $CLK/TRG_3$ に来るパルスを数える。そこには $ZC/TO_0$ がつながっているから，結局0.0128秒ごとに来るパルスを数えることになる。タイムコンスタントは200だから，1個パルスが来るたびに「200，199，198，……，2，1」となる。1の次はまた200で，延々とそれを繰り返すのである。

　ではぼちぼちと本筋へ入ってゆく。CTCの本筋といえば，当然ながらタイマ割り込みを使った「マルチタスク」である。理解に便利なように，CTCの設定をBASICで書いてある。**リスト5-2**は，「割り込み実行ルーチン」のアセンブルリスト，**リスト5-3**がそれを使った例である。

　リスト5-3をRUNすると「OK」と表示されたのち，画面に筋が走り出すであろう。しかしその他はまったく正常なBASICのコマンド待ちの状態になっているはずである。試しにリストを取ってみることもできる。その場合はグラフィックが表示されなくなるが，CTRL＋D を押せばまた表示されるはずである。これは遊びのようなサンプルであるが，タイマ割り込みを理解するにはちょうどよいだろう。

　では実際に何が起こっているのかを，**ねっとりと**解説してみる。

　まず，220行でチャンネル0に対してやっていることは，タイムコンスタントが125になったこと以外はリスト5-1の150行と同じである。次に230行で $58_H$ をOUTしているが，これは $D_0$＝0であるから割り込みベクトルの設定である。これにより，割り込みベクトルが，

　チャンネル0は $58_H$
　チャンネル1は $5A_H$
　チャンネル2は $5C_H$
　チャンネル3は $5E_H$

となる。240行にあるチャンネル3の設定は $D_7$＝1として割り込みを起こさせている点以外はこれもリスト5-1の160行と同じである。

リスト 5-2　割り込み実行ルーチン

```
                                        .Z80
                                        .PHASE  0E000H
                                ;
E000    F5                      RUNG:   PUSH    AF
E001    C5                              PUSH    BC
                                ;
E002    ED 4B E01D                      LD      BC,(GADD)
E006    78                              LD      A,B
E007    FE 40                           CP      40H
E009    30 03                           JR      NC,OKOUT        ;BC >= 4000H?
                                ;
E00B    01 4000                         LD      BC,4000H
                                ;
E00E    3A E01F                 OKOUT:  LD      A,(GPAT)
E011    ED 79                           OUT     (C),A
E013    03                              INC     BC
E014    ED 43 E01D                      LD      (GADD),BC
E018    C1                              POP     BC
E019    F1                              POP     AF
E01A    FB                              EI
E01B    ED 4D                           RETI
                                ;
E01D    4000                    GADD:   DW      4000H
E01F    FF                      GPAT:   DB      0FFH
                                ;
                                        END
```

リスト 5-3　タイマ割り込みを使って BASIC と共走(turbo BASIC では動かない)

```
100 OUT &H1FA0,3:'ﾁｬﾝﾈﾙ0 RESET
110 OUT &H1FA1,3:'ﾁｬﾝﾈﾙ1 RESET
120 OUT &H1FA2,3:'ﾁｬﾝﾈﾙ2 RESET
130 OUT &H1FA3,3:'ﾁｬﾝﾈﾙ3 RESET
140 '
150 CLEAR &HE000
160 MEM$(&HE000,16)=HEXCHR$("F5 C5 ED 4B 1D E0 78 FE 40 30 03 01 00 40 3A 1F")
170 MEM$(&HE010,16)=HEXCHR$("E0 ED 79 03 ED 43 1D E0 C1 F1 FB ED 4D 00 40 FF")
180 MEM$(&H5E,2)    =HEXCHR$("00 E0"):'ﾜﾘｺﾐ TABLE SET
190 '
200 CLS 4:INIT
210 '
220 OUT &H1FA0,&B100111   :OUT &H1FA0,125:'ﾁｬﾝﾈﾙ0 SET
230 OUT &H1FA0,&H58                      :'ﾜﾘｺﾐﾍﾞｸﾄﾙ SET
240 OUT &H1FA3,&B11000111:OUT &H1FA3,125:'ﾁｬﾝﾈﾙ3 SET
```

さて，ちょっと前に戻って 180 行を見ていただきたい。ここで $005E_H$番地からの 2 バイトの $00_H$，$E0_H$を書き込んでいる。これがチャンネル 3 の割り込みテーブルである。仕掛けはこれだけである。これでチャンネル 3 により，

$$250(ns) \times 256 \times 125 \times 125 = 1(秒)$$

ごとに割り込みが起き，そのたびに $E000_H$番地から始まるリスト 5-2 の割り込み実行ルーチンに飛んでくることになる。この機械語ルーチンは見てのとおりに，1 バイト（8 ドット）ずつグラフィックにデータを書き込んでいくだけである。220 行もしくは 240 行の「125」を「1」に書き換えると書き込みが速くなるので，CLS4 と追いかけっこするのもまた楽しからずやである。

# 実技編である

タイマの割り込みを実用的に使う目的としては次の三つが挙げられるだろう。

**①　一定の間隔（リズム）で何かをしたいとき**

たとえばturbo BASICのTEMPO文ではタイマ割り込みの間隔を変えることにより音楽の演奏速度を変えている。また，ゲームの最中に流れている音楽が，敵キャラがたくさん出ているときは遅く，逆に少ないときは速くなったりするのをよく見かけるが，タイマを使えばそのようなことを比較的簡単に避けることができる。余談になるが，機械語で作られたプログラムでは，垂直帰線期間信号や垂直同期信号(I/Oポートの$1A01_H$)を使って一定のリズムでBGMを鳴らす手法がある。具体的にどうするかというと，これらの信号は1/60秒ぐらいの周期でHigh/Lowを繰り返すので，

**頻繁にその信号を監視しておき，立ち上がりや立ち下がりを見つけることができれば，**

1/60秒のリズムを捕えるのである。実際にやっているソフトは，アルシスソフトの「ウイバーン」,「リバイバー」などである。ううむ，なんという知恵者なことよ。

**② 二つのことを同時にしたいとき**

一番よい例が同じくturbo BASICのMUSIC@文である。turbo BASICではMUSIC@文中の演奏データは（どっかの）バッファに入れておくだけで，実際の演奏はその後に続くBASICのコマンドを処理しながら，時々タイマ割り込みを使って，適当なデータをPSGに送り込むことによって，行なっている。もっと簡単に言ってしまうと，ゲームでPCGやグラフィックを動かしながら（リズムを狂わさずに）BGMや効果音を鳴らすことができる。

**③ 遅い周辺機器を待たずにCPUをフルに使いたいとき**

これは②の場合とかなり近いものだが，ひとまず区別しておく。これで一番よくあるのが（本体側ソフトウェアによる）プリンタバッファである。つまりプリンタの印字速度が遅いために，CPUがしこたま待たされるのを避けるためのものである。念のために言うが，プリンタから本体へはREADY信号というものが来ていて，結局は「本体さん，私ことプリンタにデータを1バイト送ってもいいよ」ということを示す信号なのである。この信号が「まだダメ」であるなら，CPUは待っていなければならないのだ。そこでプリンタバッファの動作を簡単に説明すると，まずはプリンタに送りたいデータはバッファに入れておく。そしてタイマ割り込みがかかるたびにREADY信号をチェックして,「送ってもいいよ」と言えば1バイト送ってやるのである。要するに「他の仕事をしながら，時々様子を見る」という使い方である。

**④ そのものズバリに時計を作る**

一定時間ごとに割り込みを起こしてくれるのだから，その回数を数えていれば時間が正確に分かるのである。短い方では，1/100秒の測定などは朝飯前である。またメモリが許す限り長いタイマも作成可能である。たとえば何年，何世紀を1/100秒単位で計ることも簡単である。

そこでまず，②のBGMからやってみる。対比のために**リスト5-4**にタイマを使わない場合も載せておく。

説明しよう。最初に並んでいるMEM$文は機械語プログラムなどではなく，演奏データである。全部BASICなのだから配列にすりゃーいいのに，と思うだろうが，私はしたいよ

リスト 5-4　BASIC 版 BGM 実行ルーチン

```
100 CLEAR &HD000
110 MEM$(&HD000,12)=HEXCHR$("00 DD 01 01 08 0F 10 05 08 00 10 05")
120 MEM$(&HD00C,12)=HEXCHR$("00 A9 01 01 08 0F 10 05 08 00 10 05")
130 MEM$(&HD018,12)=HEXCHR$("00 7B 01 01 08 0F 10 05 08 00 10 05")
140 MEM$(&HD024,12)=HEXCHR$("00 DD 01 01 08 0F 10 05 08 00 10 05")
150 MEM$(&HD030,48)=MEM$(&HD000,48):'ｸﾘｶｴｼ
160 MEM$(&HD060,12)=HEXCHR$("00 7B 01 01 08 0F 10 05 08 00 10 05")
170 MEM$(&HD06C,12)=HEXCHR$("00 65 01 01 08 0F 10 05 08 00 10 05")
180 MEM$(&HD078,12)=HEXCHR$("00 3E 01 01 08 0F 10 05 08 00 10 05")
190 MEM$(&HD084,12)=HEXCHR$("00 7B 01 01 08 0F 10 05 08 00 10 05")
200 MEM$(&HD090,48)=MEM$(&HD060,48):'ｸﾘｶｴｼ
210 DEFINT A-Z:CLICK OFF:WIDTH 40:INIT
220 X=20:Y=12:LOCATE X,Y:PRINT "A";
230 MPS=&HD000:MPE=&HD0C0:MP=MPS:W=0
240 SOUND 7,&B111000:SOUND 8,0
250 '
260 GOSUB"SOUND"
270 S=STICK(0):T=STRIG(0)
280 IF S=0 THEN FOR D=0 TO 100:NEXT:GOTO 360
290 LOCATE X,Y:PRINTSPACE$(1);:'ERASE CHAR
300 Y1=Y-((S+2)¥3)+2
310 X1=X+((S+2) MOD 3)-1
320 IF (X1>=0) AND (X1<39) THEN X=X1
330 IF (Y1>=0) AND (Y1<23) THEN Y=Y1
340 LOCATE X,Y:PRINT "A";
350 '
360 IF T THEN END
370 GOTO 260
380 '
390 LABEL"SOUND"
400 IF W THEN W=W-1:RETURN:'WAIT
410 IF MP=MPE THEN MP=MPS
420 M1=PEEK(MP):MP=MP+1:M2=PEEK(MP):MP=MP+1
430 IF M1<&H10 THEN SOUND M1,M2:GOTO 410
440 IF M1=&H10 THEN W=M2:GOTO 400
450 IF M1=&H11 THEN W=M2*256:GOTO 400
460 PRINT "DATA ERROR":STOP
```

ーにするのだ。で，演奏データの形式だが，これは 2 バイトが一組になっている。規則は簡単で，

1) 1 バイト目が $00_H$〜$0F_H$ならば，それは PSG のレジスタ番号である。PSG のその番号のレジスタに次の 1 バイトが書き込まれる。

2) 1 バイト目が $10_H$もしくは $11_H$ならば，それはウエイト（時間つぶし）コマンドである（つまり PAUSE 文のよーなもの）。$10_H$の場合は次の 1 バイトに相当する時間，PSG には何も書き込まれない（消されもしない）。$11_H$の場合は長いウエイトである。次の 1 バイト×256 に相当する時間，PSG には何も書き込まれない。

これだけじゃなんだから，110 行だけは解説しておく。まずは $00_H$，$DD_H$だから，「0 番レジスタに $DD_H$を書き込め」となる。次にある 2 組も同じよーなものである。結局この 3 組によって，チャンネル A から「ド」の音が音量 15（$08_H$，$DF_H$）で出ることになる。しかしこれだけでは音にならない。つまり，一定時間以上鳴らし続けてやらなければ，人間の耳には「ド」に聞こえないのだ（「プチッ」という音になってしまう）。そこで，次に $10_H$，$05_H$でウエイトをおいてやる。その後一度チャンネル A の音量を 0 にして，もう一度ウエイトがある。これは MUSIC 文を使ってメロディーを鳴らしたことがある人なら分かるように，音符と音符の間を区切るためのもの，つまりスタッカートである。リスト 5-4 のデー

タの仕組みは大体こんなふうになっている。

さて，このプログラムは実にセコくて，極めて原始的なことしかしていない。つまり，RUN すると画面の中央に「A」が表示され，「ドレミドードレミドーミファソミーミファソミー」というセコイ BGM がエンエンと繰り返される。しかし，お立ち会い。テンキーによって，「A」は上下左右に動くが，BGM はほとんど乱れずに続くのである。

さっさとネタをばらしてしまうが，これは「キャラクタを動かす」というプログラムがループになっているということを利用しているのだ。どーゆーことかというと，まず,「ド」を出すならば，PSG をそのよーに設定してやる。つまり PSG のレジスタの 0，1，8 にそれぞれの値を書き込んでやる。〝SOUND〟に GOSUB すると，ウエイト中かもしくはウエイトコマンドが見つかるまでは RETURN しないので，最初に〝SOUND〟に GOSUB した段階で，すでに PSG は「ド」の音を出しているのである。

ウソだと思ったら（思うかな？）265 行に「END」を入れてみるといい。「ド」が鳴り続くはずである(止めるときは CTRL + D )。〝SOUND〟から RETURN した後は，キャラクタを動かす作業だけをすればよい。ただし，ループの時間（次に〝SOUND〟に GOSUB するまでの時間）を，あまり変えないようにすること。これは 280 行を見てもらえば分かると思うが，テンキーからの入力がなくてキャラクタを動かす必要がない場合は，FOR 文で時間をつぶしている。その他の 290〜340 行はキャラクタを動かしているだけのルーチンである。290 行は「A」を消している部分，300，310 行はよくあるテクニックを使った新しい位置の計算である。320，330 行は画面からはみ出さないようにチェックしてから，新しい位置を X，Y に代入している。340 行はその位置への表示である。

では，2 回目に〝SOUND〟に GOSUB したときはどうなっているかを見てみる。まず，前回のときはウエイトコマンドを見つけて，その後 400 行で W＝W－1 とした後 RETURN したのである（分かるかな？）。それは 440 行である。変数 W に 2 バイト目が代入された後に，さらに－1 されたのだから，2 回目に GOSUB してきた時点では W の値は 4 である。すると，400 行の「IF W〜」に引っかかる。すなわち W から 1 引いて，W の値を 3 にしてから RETURN である。これと同じことが W が 0 になるまで繰り返されるわけだから，結局，

1 回目→ PSG は「ド」を出し始める（W＝4 で RETURN)
2 回目→何もしない（W＝3 で RETURN)
3 回目→何もしない（W＝2 で RETURN)
4 回目→何もしない（W＝1 で RETURN)
5 回目→何もしない（W＝0 で RETURN)
6 回目→ W＝0 だから，次のコマンドを捜す

ということになる。1 回目と 2 回目，2 回目と 3 回目，……の間には，270〜370 行を実行している時間がそれぞれ挟み込まれているから，この場合，数えてみるとループ 5 回分の間「ド」が出ることになる。よーく考えると分かることだが，440 行，450 行で「GOTO 400」とせずに「RETURN」とすると，ループ 6 回分になってしまうので，この手のプログラムを作るときは注意が必要である。

残りの部分を種明しすると，MP は演奏データ（アドレス）を指している変数である。MPS（データの始まり），MPE（データの終わりの次）は，BGM を繰り返すためのもので，MP が MPE と等しくなったら，MP に MPS を代入してやって繰り返しに入るのである。後は 240 行であるが，これは本来ならば $D000_H$ からの音楽データの先頭に，

$07_H$，$28_H$，$08_H$，$00_H$

というように入れてやるべきだったのだが，アドレスがズレてしまって美しくなくなるので手抜きをしたのだ。ま，以上である。

すでに気付いたことと思うが，このテクニックを使った場合，BASIC だろうが機械語だろうが，キャラクタを動かすルーチンの部分が多少なりとも複雑になってしまうと，BGM のリズム（テンポ）を一定に保つのは至難の技なのだ。まず第一に，ある局面になったら新しく敵キャラが出てくる場合や，さらにその敵キャラが増減する場合である。その度に，うまくヒマつぶしの時間を増減させてやらなければならないのだ。実にうっとうしい話ではないか。

**そこで CTC 様がリングに上がるのである。**タイマ割り込みによって変わるのは次の 2 点である。

1） GOSUB "SOUND" をループの中に置かなくてよい（決められた時刻になると，自動的に GOSUB してしまう）。

2） テンポを一定に保つためのヒマつぶしのルーチンを考えなくてもよい。

なんだなんだ！　結局は「音無しのゲームを普通に作る」だけで済んでしまうのではないかっ！　ということで**リスト 5-5，5-6** である。リスト 5-5 はリスト 5-4 のプログラムをそのまま機械語に置き換えたようなものであるから説明はしない。

**リスト 5-5　BGM 用割り込み実行ルーチン**

```
                                  .Z80
                                  .PHASE   0E000H
                          ;
 E000     F5              PLAY:   PUSH     AF
 E001     C5                      PUSH     BC
 E002     D5                      PUSH     DE
 E003     E5                      PUSH     HL
                          ;
 E004     2A E051         PLAY0:  LD       HL,(WC)
 E007     7C                      LD       A,H
 E008     B5                      OR       L
 E009     28 06                   JR       Z,NOWAIT
 E00B     2B                      DEC      HL
 E00C     22 E051                 LD       (WC),HL
 E00F     18 39                   JR       PEND
                          ;
 E011     2A E057         NOWAIT: LD       HL,(MPE)
 E014     ED 5B E053              LD       DE,(MP)
 E018     B7                      OR       A
 E019     ED 52                   SBC      HL,DE    ;MPE>MP?
 E01B     38 02                   JR       C,OVER
 E01D     20 04                   JR       NZ,FETCH
 E01F     ED 5B E055      OVER:   LD       DE,(MPS)
                          ;
 E023     EB              FETCH:  EX       DE,HL    ;HL=MP
 E024     7E                      LD       A,(HL)
 E025     23                      INC      HL
 E026     5E                      LD       E,(HL)
 E027     23                      INC      HL
 E028     22 E053                 LD       (MP),HL ;STORE MP
                          ;
 E02B     FE 10                   CP       16
```

```
E02D     30 0A                   JR      NC,SETWT ;SET WAIT COUNTER
                         ;
E02F     01 1C00                 LD      BC,1C00H
E032     ED 79                   OUT     (C),A    ;SET REG. NOMBER
E034     05                      DEC     B
E035     ED 59                   OUT     (C),E
E037     18 D8                   JR      NOWAIT
                         ;
E039     FE 10           SETWT:  CP      16
E03B     20 04                   JR      NZ,SETWT1        ;BIG WAIT
E03D     16 00                   LD      D,0
E03F     18 03                   JR      SETWT2
E041     53              SETWT1: LD      D,E
E042     1E 00                   LD      E,0
E044     ED 53 E051      SETWT2: LD      (WC),DE
E048     18 BA                   JR      PLAY0
                         ;
E04A     E1              PEND:   POP     HL
E04B     D1                      POP     DE
E04C     C1                      POP     BC
E04D     F1                      POP     AF
E04E     FB                      EI
E04F     ED 4D                   RETI
                         ;
E051     0000            WC:     DW      0000H    ;WAIT COUNTER
E053     0000            MP:     DW      0000H    ;MUSIC DATA POINTER
E055     0000            MPS:    DW      0000H    ;MUSIC START
E057     0000            MPE:    DW      0000H    ;MUSIC END
                         ;
                                 END
```

リスト 5-6　機械語版 BGM 実行ルーチン(turbo BASIC では動かない)

```
100 '100 - 130 ﾆﾊ, ﾘｽﾄ5-1 ﾉ 100 - 130 ｷﾞｮｳ ｦ ｿﾉﾏﾏ ﾓｯﾃ ｸﾙ.
140 '
150 CLEAR &HE000
160 MEM$(&HE000,16)=HEXCHR$("F5 C5 D5 E5 2A 51 E0 7C B5 28 06 2B 22 51 E0 18")
170 MEM$(&HE010,16)=HEXCHR$("39 2A 57 E0 ED 5B 53 E0 B7 ED 52 38 02 20 04 ED")
180 MEM$(&HE020,16)=HEXCHR$("5B 55 E0 EB 7E 23 5E 23 22 53 E0 FE 10 30 0A 01")
190 MEM$(&HE030,16)=HEXCHR$("00 1C ED 79 05 ED 59 18 D8 FE 10 20 04 16 00 18")
200 MEM$(&HE040,16)=HEXCHR$("03 53 1E 00 ED 53 51 E0 18 BA E1 D1 C1 F1 FB ED")
210 MEM$(&HE050,16)=HEXCHR$("4D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00")
220 '
230 '230 - 330 ﾆﾊ, ﾘｽﾄ5-4 ﾉ 100 - 200 ｷﾞｮｳ ｦ RENUMBER ｼﾃ ﾓｯﾃ ｸﾙ.
340 '
350 MEM$(&HE051,8)=HEXCHR$("00 00 00 D0 00 D0 C0 D0")
360 MEM$(&H5E,2)  =HEXCHR$("00 E0")
370 '
380 SOUND 7,&B111000:SOUND 8,0
390 OUT &H1FA0,&B111     :OUT &H1FA0,100:'ﾁｬﾝﾈﾙ0 SET
400 OUT &H1FA0,&H58                     :'ﾜﾘｺﾐﾍﾞｸﾄﾙ SET
410 OUT &H1FA3,&B11000111:OUT &H1FA3,25 :'ﾁｬﾝﾈﾙ3 SET
```

# 結論である

以上なわけである。この章のサンプルは，CTC の I/O アドレスを $1FA8_H$〜$1FAB_H$にずらしてやれば，CZ-8BM2 を装着した X1 でも動くはずである。ぜひとも試していただきたい。また，リスト 5-6 では PSG のチャンネル B，C を使っていないので，第 10 章のリスト 10-3 を少々変えて，チャンネル A をいじらないようにすると，BGM と銃声が共存共栄するのである。具体的にどうするかというと，リスト 10-3 の 120 行にある「&B000111」を「&H001110」にすればよいだけなのだ。これでチャンネル A はトーン，B と C はノイズになる。**どーだ，すごいだろう。**止めるときは，「OUT &H1FA3, 3」である。

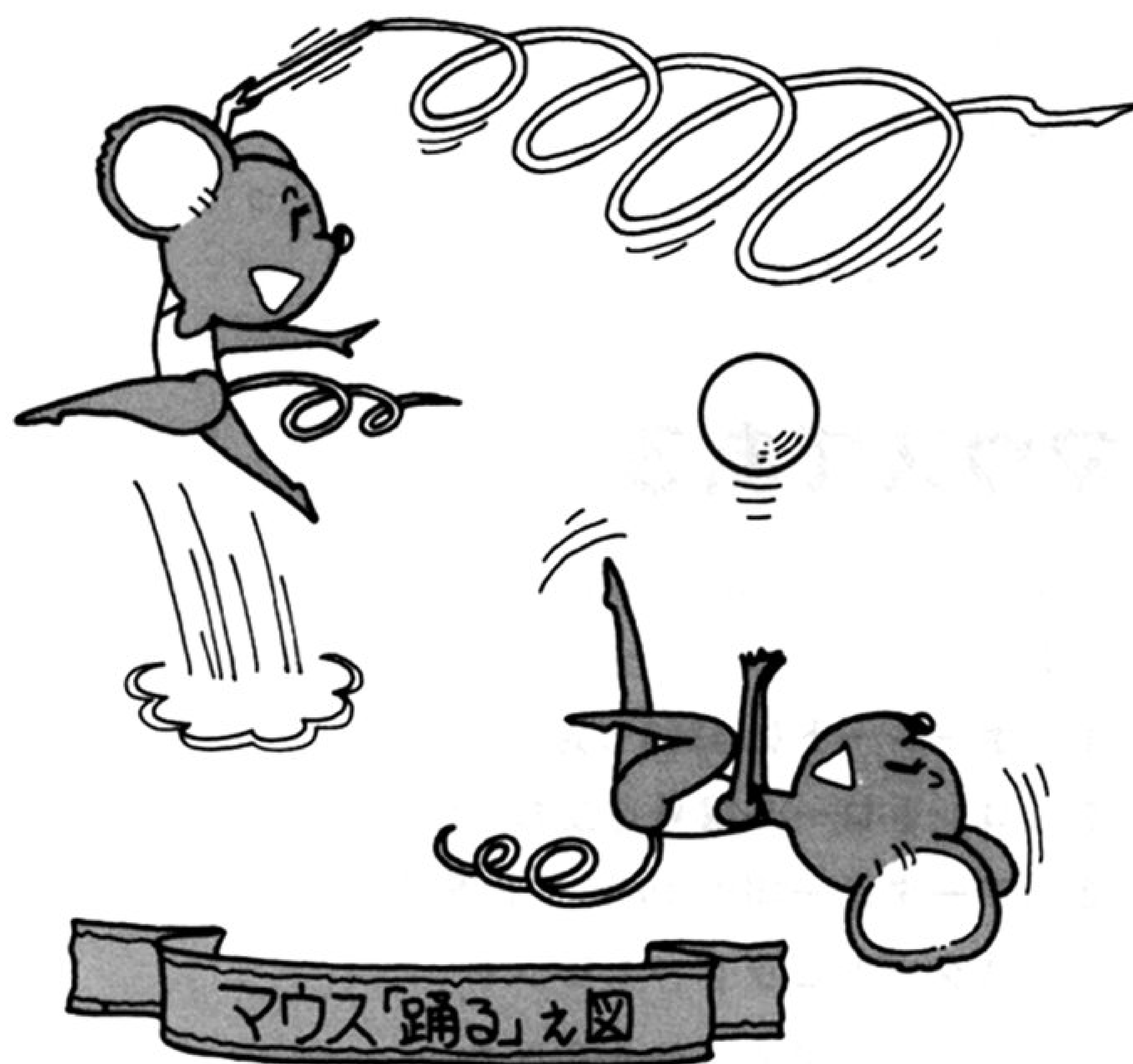
あと CTC の使い方としては時計が代表的なものであるが，以上のことを理解できればあっという間に作れてしまうだろう。

なお，あまり勧められた手ではないが，turbo BASICではちょちょいのちょいで簡単にCTCの割り込みを使う手がある。それは，$F81E_H$にある割り込みベクトルを書き換えるという手である。割り込み間隔はTEMPO文で変更できる。この場合もう一工夫しないとPLAY文が使えなくなってしまう，などの欠点があるが，手軽さには捨てがたいものがある。実例は12章（カラーイメージボード）に載せてある。

CTCの使い方は限られているが，その応用例は広いのである。そして，CTCだけではどうにもならないが，大事な基本技として必修なのである。

# 第6章

# SIO




SIOでマウスである

## 第6章

# SIOでマウスである…………

この章で取り上げるZ80 SIO (Serial Input/Output) は，シリアル通信を主な業務としている石である。そして，この石はやたらと複雑で強力なのである。これはとりもなおさず，シリアル通信というものが複雑怪奇であることに由来する。実際にパソコンで通信をする場合には，BASICのOPEN文の通信パラメータで指定するだけの機能があれば済むのだが，SIOにはそれ以上の機能が盛り込まれているのである。そういうわけで，**不本意ながら**SIOの使い方を細かく説明するわけにはいかない。もしもSIOの機能をみっちりと知りたいならば，参考文献4を読んでいただきたい。

さて，実はZ80 SIOといっても三つのバージョン（ボンディングオプション）があるのだ。すなわち，

SIO/0，SIO/1，SIO/2

である。X1に使われているのはそのうちのSIO/0である。これら三つのものの違いは，チャンネルBに対する制限で，早い話がピン数（足の数）に限りがあるのでそうなっているのである（X1ではチャンネルBはマウスに接続するわけだから，まったく問題ない）。

SIOにはチャンネルA，Bの二つのチャンネルがあり，X1ではチャンネルAがRS-232Cの通信用で，チャンネルBが今書いたようにマウス用になっている。そして，まずはこの章でチャンネルB＝マウスを先にやった後で，次の第7章でチャンネルAの通信関係へと進むつもりである。

## マウスである

X1のマウスはRS-232Cのシリアルマウスで，データの方向はマウス→X1の一方通行である。マウスから送ってくるデータは，

**① ボーレートは4800 bps**

**② コントロール信号を立ち下げる（H→L）とデータを送ってくる**

**③ データは一組が10ビット構成で，**

**スタートビット＝1ビット**

**データビット　＝8ビット**

**ストップビット＝1ビット**

**の10ビットが3組**

となっている。③でいきなりスタートビットとかストップビットとかが出てきたが，これがシリアル通信のうっとーしいところである。具体的にどういうシロモノかというと，「はい，今からデータが行きますよー」，「はい，これでデータは終わりです」とかいう区切りなのだ。一般の通信では，データが7ビットだったり，ストップビットが1.5もしくは2

図 6-1　マウスからのデータの内容

| | | | | |
|---|---|---|---|---|
| 1バイト目 | ステータス | $D_0$ | スイッチ1の状態 | 1でON，0でOFF |
| | | $D_1$ | スイッチ2の状態 | |
| | | $D_2$ | —— | |
| | | $D_3$ | —— | |
| | | $D_4$ | Xのオーバーフロービット(128以上のとき) | |
| | | $D_5$ | Xのアンダーフロービット(−129以下のとき) | |
| | | $D_6$ | Yのオーバーフロービット(128以上のとき) | |
| | | $D_7$ | Yのアンダーフロービット(−129以下のとき) | |
| 2バイト目 | X方向の移動量 | −128～+127 | | |
| 3バイト目 | Y方向の移動量 | −128～+127 | | |

$D_7$が符号ビットである
読み出した1バイトのデータがXならば，
$D_7=0$→移動量$=X\geqq 0$
$D_7=1$→移動量$=X-256<0$

ビット（長さが1.5倍もしくは2倍なのだ）のこともある。さて，結局マウスからは3バイトのデータを送ってくるわけである。その内訳は**図 6-1**である。

まずステータスについて説明する。$D_0$，$D_1$ のスイッチは自明だからよいだろう。2ビット空いて $D_4$～$D_7$がオーバー/アンダーフロービットである。これはマウス内のカウンタがあふれたということで，たとえば$D_4=1$だったなら，マウスが「おいらはよく分かんないけど，X方向に128以上動いちゃった。だから2番目のデータの『X方向の移動量』はあてにならないよ」と言っているわけである。以下同様。

次にX/Y方向の移動量であるが，一番大事なことは，前回データを送ったときからの相対距離だということである。turbo BASICのMOUSE(7)，MOUSE(8)に相当するものだから，BASICのマニュアルを見ておくれ。

では**リスト 6-1**によって，具体的にマウスを使う方法を示す。恐ろしいことにオールBASICである。心得のある人ならば「4800 bpsで30ビットを送ってくるんだから，30÷4800=0.00625秒じゃねーか。BASICで間に合うの？」と思うであろう。しかし目出たいことに，Z80 SIOはバッファを三つ分（この場合だと3バイト分）持っているのだ。だからどんなにゆっくりやっても，マウスからのデータならば取りこぼしがないのである。楽ちん楽ちん。

リスト 6-1　マウスの使い方，もしくはバッファは3拍子

```
100 GOSUB"SETCTC&SIO"
110 '
120 OUT &H1F93,5:OUT &H1F93,0   :'ｺﾝﾄﾛｰﾙ ｼﾝｺﾞｳ ｦ H ﾆ ｽﾙ
130 PAUSE 0                     :'ﾁｮｯﾄ ﾀﾞｹ ﾏﾂ (ｷﾌﾞﾝ)
140 OUT &H1F93,5:OUT &H1F93,2   :'REQUEST(ｺﾝﾄﾛｰﾙ ｼﾝｺﾞｳ=RTS ﾉ ﾀﾁｻｹﾞ)
150 FOR I=1 TO 3
160   OUT &H1F93,0:BBB=INP(&H1F93) AND 1 :'BUFF ﾆ ﾃﾞｰﾀ ｶﾞ ｱﾙｶ ?
170   IF BBB=0 THEN 160
180   DDD=INP(&H1F92)
190   PRINT DDD
200 NEXT
```

```
210 PRINT STRING$(10,"=")
220 GOTO 120
230 '
240 LABEL"SETCTC&SIO"
250 OUT &H1FA2,&B1000111:'CTC ﾁｬﾝﾈﾙ 3 SET
260 OUT &H1FA2,26       :'TIME CONSTANT
270 '
280 RESTORE "SIODATA"
290 FORI=1TO15
300   READ D:OUT&H1F93,D
310 NEXT
320 RETURN
330 '
340 LABEL"SIODATA"
350 DATA   &B00011000   :'RESET
360 DATA 1,0            :'WR0=00H
370 DATA 2,00H          :'WR2=00H=INT. VECTOR (turbo ﾉ ﾊﾞｱｲ)
380 DATA 4,&B01000100   :'WR4=44H='ｸﾛｯｸ ﾊ *16' & 'ｽﾄｯﾌﾟﾋﾞｯﾄ=1 ﾋﾞｯﾄ'
390 DATA 5,00H          :'WR5=00H
400 DATA 6,00H          :'WR6=00H
410 DATA 7,00H          :'WR7=00H
420 DATA 3,&B11000001   :'WR3=C1H='DATA=8ﾋﾞｯﾄ' & 'ｲﾈｰﾌﾞﾙ ﾚｼｰﾊﾞ'
```

さて，リスト 6-1 はしょっぱなから 240 行へ GOSUB している。これは CTC と SIO の設定である。まずは 250～260 行で SIO のチャンネル B 用のクロックを作っている CTC のチャンネル 2 を，「カウンタモードでタイムコンスタントは 26」にする。これによってチャンネル 2 は 2MHz のクロックを数えるわけだから，

$$\frac{1}{2\times10^{6}(\mathrm{Hz})}\times26=13\times10^{-6}(\text{秒})$$

ごとにパルスを出す。SIO は，このパルスを 16 もしくは 64 分周してボーレートにするのである。この場合は，16 分周すれば，

$$13\times10^{-6}\times16=208\times10^{-6}(\text{秒})$$

となり，結局これは，

$$\frac{1}{208\times10^{-6}}=4807.69\cdots\fallingdotseq4800$$

というしだいである。きっちり 4800 である必要はないのである。次は 280 行からであるが，ここで SIO を設定している。やっていることは，

- キャラクタ長は 8 ビット
- クロックは CTC からのパルスを 16 分周
- ストップビットは 1
- 割り込みなし

などである。

ここを RETURN した後に，いよいよメインルーチンである。120～140 行ではコントロール信号を立ち下げているのだが，マウスへのコントロール信号はチャンネル B の RTS なのである。RTS とは「Request To Send」で，要するに「送ってきてちょーだい」である。これが H → L になると，マウスは律儀に 3 バイトのデータを送ってくる。130 行に PAUSE 文があるが，これは気分の問題である。BASIC ならばどうでもよいのだが，機械語でやる場合はそうはいかない。ある程度の時間「H」にしておいてから「L」にしなけれ

ばならないのだ。実にやっかいなものだ。

150〜200行のFOR〜NEXT文で3バイトのデータをSIOのバッファから読み出して表示している。160，170行は「バッファにデータが入っているか」の判定ルーチン，180行がデータの読み出しである。プログラムをRUNすると3バイトのデータを表示していく。マウスを動かしたり，ボタンを押したりすると数字が変わるであろう。

# マウスの実技である

悲しいことであるが，世の中には「マウスよりもトラックボールがいいのだ」，「うんにゃ，ライトペンこそが一等賞！」，「てやんでぃ。タッチパネルを忘れてもらっちゃ困るぜぃ」などと言う人達がいる。あなた達に**神の御許し**がありますよーに，というわけで私はマウス党なのである。

そこで**リスト6-3，6-4**をさっさと実行していただきたい。ちなみにリスト6-3のソースリストは**リスト6-2**となっている。リスト6-3はチェックサムとCRC付きのダンプリストである。入力に際しては付録Aのダンプリストチェックプログラムを参照のこと。

**リスト6-2　マウスドライバ，もしくはプリマドンナ**

```
                                .Z80
                                .PHASE  0EA00H
                        ;
1F90                    ZSIO    EQU     1F90H
1FA0                    ZCTC    EQU     1FA0H
0018                    PSIZE   EQU     14+8+2          ;PARAMETER SIZE
                        ;
EA00    EB              MOUSE:  EX      DE,HL
EA01    5E                      LD      E,(HL)
EA02    23                      INC     HL
EA03    56                      LD      D,(HL)
EA04    EB                      EX      DE,HL
                        ;
EA05    22 EB29                 LD      (RETVA),HL
                        ;SAVE RETURN VALUE AREA
EA08    11 EB31                 LD      DE,X1
EA0B    01 0018                 LD      BC,PSIZE
EA0E    ED B0                   LDIR                    ;COPY
                        ;
EA10    3E 50                   LD      A,50H
EA12    32 EC61                 LD      (PATCH1+1),A
EA15    3E 29                   LD      A,29H           ;ADD HL,HL
EA17    32 EC7A                 LD      (PATCH2),A      ;SET WIDTH 80
EA1A    3E 28                   LD      A,28H           ;28H=JR Z,
EA1C    32 EAAC                 LD      (PATCH3),A
EA1F    21 ECA6                 LD      HL,ARW8         ;W 80 CURSOR
EA22    22 EB2F                 LD      (DFMC),HL       ;DEFAULT CURSOR
EA25    3A EB48                 LD      A,(MODE)
EA28    0F                      RRCA                    ;CHECK BIT0
EA29    38 17                   JR      C,WI80
                        ;
EA2B    47                      LD      B,A
EA2C    3E 28                   LD      A,28H           ;DATA
EA2E    32 EC61                 LD      (PATCH1+1),A
EA31    3E 00                   LD      A,000H          ;NOP
EA33    32 EC7A                 LD      (PATCH2),A
EA36    3E 18                   LD      A,18H           ;18H=JR
EA38    32 EAAC                 LD      (PATCH3),A
EA3B    21 EDD4                 LD      HL,ARW4
EA3E    22 EB2F                 LD      (DFMC),HL
EA41    78                      LD      A,B
                        ;
EA42    0F              WI80:   RRCA
EA43    30 04                   JR      NC,MSGO
EA45    CD EBD9                 CALL    PSET            ;JUST GRAPHIC
```

```
EA48    C9                       RET
                         ;
EA49    2A EB31          MSGO:   LD      HL,(X1)         ;COPY POSITION
EA4C    22 EB2B                  LD      (X0),HL
EA4F    2A EB33                  LD      HL,(Y1)
EA52    22 EB2D                  LD      (Y0),HL
EA55    CD EADC                  CALL    CIN             ;CHECK IN
EA58    E6 03                    AND     3
EA5A    28 05                    JR      Z,BEGIN
                         ;
EA5C    3E 02                    LD      A,2
EA5E    C3 EB16                  JP      ERR0
                         ;
EA61    CD EBA5          BEGIN:  CALL    SETMS           ;MOUSE INIT
EA64    CD EBD9                  CALL    PSET            ;SHOW CURSOR
                         ;
EA67    FD 21 EB39               LD      IY,MSDATA
EA6B    CD EB49                  CALL    MSIN
                         ;
EA6E    3A EB39          BEGIN0: LD      A,(MSDATA)
EA71    F5                       PUSH    AF
EA72    FD 21 EB39               LD      IY,MSDATA       ;IY=POINTER
EA76    CD EB49                  CALL    MSIN            ;READ MOUSE
EA79    F1                       POP     AF
EA7A    B7                       OR      A
EA7B    20 11                    JR      NZ,BUT2
                         ;
EA7D    21 EB35                  LD      HL,BMASK        ;CONDITION
EA80    3A EB39                  LD      A,(MSDATA)      ;GET STAT
EA83    E6 03                    AND     3
EA85    06 04                    LD      B,4
                         ;
EA87    BE               BUT0:   CP      (HL)            ;CHECK BUTTON
EA88    23                       INC     HL
EA89    CA EB11                  JP      Z,BUTTON        ;BUTTON HIT
EA8C    10 F9            BUT1:   DJNZ    BUT0
                         ;
EA8E    21 EB3A          BUT2:   LD      HL,MSDATA+1
EA91    7E                       LD      A,(HL)
EA92    23                       INC     HL
EA93    B6                       OR      (HL)
EA94    28 D8                    JR      Z,BEGIN0        ;X=Y=0 THEN NO-O
P
                         ;
EA96    3A EB3A                  LD      A,(MSDATA+1)
EA99    CD EB0A                  CALL    XSIGN
EA9C    2A EB31                  LD      HL,(X1)
EA9F    19                       ADD     HL,DE
EAA0    22 EB2B                  LD      (X0),HL
                         ;
EAA3    3A EB3B                  LD      A,(MSDATA+2)
EAA6    CD EB0A                  CALL    XSIGN
EAA9    3E 01                    LD      A,1
EAAB    BB                       CP      E
EAAC    28 04            PATCH3: JR      Z,OK1   ;1 -> 1 (-1 -> -1)
                         ;OR      JR      OK1      (WIDTH 40)
EAAE    CB 2A                    SRA     D
EAB0    CB 1B                    RR      E       ;DE=DE/2 with sign
EAB2    2A EB33          OK1:    LD      HL,(Y1)
EAB5    19                       ADD     HL,DE
EAB6    22 EB2D                  LD      (Y0),HL
                         ;
                         ;CHRCK IN?
EAB9    CD EADC                  CALL    CIN
EABC    FE 03                    CP      3
EABE    28 AE                    JR      Z,BEGIN0        ;X AND Y OUT
                         ;
EAC0    F5                       PUSH    AF
EAC1    CD EBD9                  CALL    PSET            ;ERASE CURSOR
EAC4    F1                       POP     AF
                         ;
EAC5    0F                       RRCA
EAC6    38 06                    JR      C,YCHECK        ;X=OB THEN CHECK
 Y
EAC8    2A EB2B                  LD      HL,(X0)         ;COPY POSITION
EACB    22 EB31                  LD      (X1),HL
                         ;
EACE    0F               YCHECK: RRCA
EACF    38 06                    JR      C,SHOWC
EAD1    2A EB2D                  LD      HL,(Y0)
```

```
EAD4    22 EB33                 LD      (Y1),HL
                        ;
EAD7    CD EBD9         SHOWC:  CALL    PSET            ;SHOW NEW CURSOR
EADA    18 92                   JR      BEGIN0  ;AGAIN
                        ;
EADC    AF              CIN:    XOR     A
EADD    ED 5B EB44              LD      DE,(YU)
EAE1    2A EB2D                 LD      HL,(Y0)
EAE4    B7                      OR      A
EAE5    ED 52                   SBC     HL,DE
EAE7    38 09                   JR      C,XIN
                        ;
EAE9    ED 5B EB2D              LD      DE,(Y0)
EAED    2A EB46                 LD      HL,(YD)
EAF0    ED 52                   SBC     HL,DE
                        ;
EAF2    17              XIN:    RLA
EAF3    ED 5B EB40              LD      DE,(XL)
EAF7    2A EB2B                 LD      HL,(X0)
EAFA    B7                      OR      A
EAFB    ED 52                   SBC     HL,DE
EAFD    38 09                   JR      C,XIN0
                        ;
EAFF    ED 5B EB2B              LD      DE,(X0)
EB03    2A EB42                 LD      HL,(XR)
EB06    ED 52                   SBC     HL,DE
EB08    17              XIN0:   RLA
EB09    C9                      RET
                        ;
EB0A    5F              XSIGN:  LD      E,A
EB0B    16 00                   LD      D,0
EB0D    07                      RLCA
EB0E    D0                      RET     NC
EB0F    15                      DEC     D       ;D=FFH or 00H
EB10    C9                      RET
                        ;
EB11    AF              BUTTON: XOR     A
EB12    18 02                   JR      ERR0
                        ;
EB14    3E 01           ERR:    LD      A,1
EB16    32 EB3C         ERR0:   LD      (ECODE),A
EB19    CD EBD9                 CALL    PSET            ;ERASE CURSOR
EB1C    ED 5B EB29              LD      DE,(RETVA)
EB20    21 EB31                 LD      HL,X1
EB23    01 0018                 LD      BC,PSIZE
EB26    ED B0                   LDIR
EB28    C9                      RET
                        ;
EB29                    RETVA:  DS      2       ;RETURN VALUE ADD.
EB2B                    X0:     DS      2       ;BUFF.
EB2D                    Y0:     DS      2       ;BUFF.
EB2F                    DFMC:   DS      2       ;DEFAULT CURSOR
                        ;
EB31                    X1:     DS      2       ;X POSITION
EB33                    Y1:     DS      2       ;Y POSITION
EB35                    BMASK:  DS      4       ;BUTTON CONDITION
EB39                    MSDATA: DS      3       ;MOUSE DATA
EB3C                    ECODE:  DS      1       ;ERROR CODE
EB3D                    MC:     DS      2       ;MOUSE CURSOR PAT ADD
EB3F                    MCCOL:  DS      1       ;MOUSE CURSOR COLOR
EB40                    XL:     DS      2       ;X UNDER LIMIT
EB42                    XR:     DS      2       ;X UPPER LIMIT
EB44                    YU:     DS      2       ;Y UNDER LIMIT
EB46                    YD:     DS      2       ;Y UPPER LIMIT
EB48                    MODE:   DS      1       ;WIDTH AND WRITE PAT
                        ;
EB49    01 1F93         MSIN:   LD      BC,ZSIO+3
EB4C    3E 05                   LD      A,5     ;->WR5
EB4E    ED 79                   OUT     (C),A   ;WR5 SELECT
EB50    AF                      XOR     A
EB51    ED 79                   OUT     (C),A   ;RTS High
                        ;
EB53    16 80                   LD      D,80H
EB55    15              MSINW:  DEC     D       ;WAIT
EB56    20 FD                   JR      NZ,MSINW
                        ;
EB58    3E 05                   LD      A,5
EB5A    ED 79                   OUT     (C),A   ;WR5 SELECT
EB5C    3E 02                   LD      A,2
EB5E    ED 79                   OUT     (C),A   ;RTS Low
```

```
                                ;
EB60    1E 05                           LD      E,5         ;RETRY COUNT
EB62    FD 22 EB80                      LD      (IYBUFF),IY
EB66    16 03                   MSRTRY: LD      D,3         ;COUNT
                                ;
EB68    CD EB82                 MSINL:  CALL    MSGET
EB6B    38 09                           JR      C,FAIL
EB6D    FD 77 00                        LD      (IY+0),A
EB70    FD 23                           INC     IY
EB72    15                              DEC     D
EB73    20 F3                           JR      NZ,MSINL
EB75    C9                              RET
                                ;
EB76    FD 2A EB80              FAIL:   LD      IY,(IYBUFF)
EB7A    1D                              DEC     E
EB7B    20 E9                           JR      NZ,MSRTRY
                                ;
EB7D    C3 EB14                         JP      ERR
                                ;
EB80                            IYBUFF: DS      2
                                ;
EB82    01 1F93                 MSGET:  LD      BC,ZSIO+3
EB85    21 0000                         LD      HL,0000H
EB88    2B                      MSGET0: DEC     HL
EB89    7C                              LD      A,H
EB8A    B5                              OR      L
EB8B    28 0D                           JR      Z,LATE
                                ;
EB8D    AF                              XOR     A
EB8E    ED 79                           OUT     (C),A       ;RR0 SELECT
EB90    ED 78                           IN      A,(C)
EB92    1F                              RRA                 ;Bit0 ON?
EB93    30 F3                           JR      NC,MSGET0
                                ;
EB95    0B                              DEC     BC          ;BC=1F92H
EB96    ED 78                           IN      A,(C)       ;GET DATA
EB98    B7                              OR      A           ;RESET CARRY FLAG
EB99    C9                              RET
                                ;
EB9A    D5                      LATE:   PUSH    DE
EB9B    CD EBA5                         CALL    SETMS
EB9E    D1                              POP     DE
                                ;
                                ;HL=0000H
EB9F    2B                      LATE0:  DEC     HL
EBA0    7C                              LD      A,H
EBA1    B5                              OR      L
EBA2    20 FB                           JR      NZ,LATE0
                                ;
EBA4    C9                              RET
                                ;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;
EBA5    F3                      SETMS:  DI
EBA6    CD EBAE                         CALL    SETCTC
EBA9    CD EBBA                         CALL    SETSIO
EBAC    FB                              EI                  ;INITIALIZE
EBAD    C9                              RET
                                ;
EBAE    01 1FA2                 SETCTC: LD      BC,ZCTC+2           ;CTC2=CHANNEL B
EBB1    3E 47                           LD      A,01000111B         ;MODE
EBB3    ED 79                           OUT     (C),A
EBB5    3E 1A                           LD      A,26                ;TIME CONSTANT
EBB7    ED 79                           OUT     (C),A
EBB9    C9                              RET
                                ;
EBBA    01 1F93                 SETSIO: LD      BC,ZSIO+3           ;CHANNEL B
EBBD    21 EBCA                         LD      HL,SIODAT
EBC0    16 0F                           LD      D,15                ;COUNT
EBC2    7E                      SIOL:   LD      A,(HL)
EBC3    23                              INC     HL
EBC4    ED 79                           OUT     (C),A
EBC6    15                              DEC     D
EBC7    20 F9                           JR      NZ,SIOL
                                ;
EBC9    C9                              RET
                                ;
EBCA    18                      SIODAT: DB      00011000B
EBCB    01 00                           DB      1,00H
EBCD    02 70                           DB      2,70H
EBCF    04 44                           DB      4,01000100B
```

```
 EBD1     05 00                         DB      5,00H
 EBD3     06 00                         DB      6,00H
 EBD5     07 00                         DB      7,00H
 EBD7     03 C1                         DB      3,11000001B
                               ;
                               ;HL=Y1,BC=X1,DE=addr
 EBD9     ED 4B EB31           PSET:    LD      BC,(X1)
 EBDD     2A EB33                       LD      HL,(Y1)
 EBE0     ED 5B EB3D                    LD      DE,(MC)
 EBE4     7A                            LD      A,D
 EBE5     B3                            OR      E        ;DE=0000 ?
 EBE6     20 04                         JR      NZ,PSET1
 EBE8     ED 5B EB2F                    LD      DE,(DFMC)        ;DEFAULT CURSOR
 EBEC     3A EB3F              PSET1:   LD      A,(MCCOL)
 EBEF     CD EBF8                       CALL    PAT
 EBF2     C9                            RET
                               ;
 EBF3                          PATD:    DS      2
 EBF5                          PATB:    DS      2
 EBF7                          COLOR:   DS      1
                               ;
                               ;HL=Y1,BC=X1,DE=PATTERN ADD.
                               ;
 EBF8     ED 53 EBF3           PAT:     LD      (PATD),DE        ;SRORE PAT. ADD.
 EBFC     0F                            RRCA
 EBFD     0F                            RRCA
 EBFE     32 EBF7                       LD      (COLOR),A        ;STORE COLOR
                               ;
 EC01     CD EC6E                       CALL XYADDR
 EC04     22 EBF5                       LD      (PATB),HL        ;STORE BEGIN ADD
R
                               ;discard MASK
 EC07     6A                            LD      L,D
 EC08     26 00                         LD      H,0
 EC0A     29                            ADD     HL,HL
 EC0B     ED 5B EBF3                    LD      DE,(PATD)
 EC0F     19                            ADD     HL,DE
 EC10     11 EBF3                       LD      DE,PATD
 EC13     01 0002                       LD      BC,2
 EC16     ED B0                         LDIR
                               ;
 EC18     2A EBF3              PATGO:   LD      HL,(PATD)
 EC1B     ED 4B EBF5                    LD      BC,(PATB)
 EC1F     78                            LD      A,B
 EC20     E6 C0                         AND     0C0H
 EC22     5F                            LD      E,A      ;E=40H,80H or C0H
 EC23     57                            LD      D,A
 EC24     E2 EC29                       JP      PO,PATGO1        ;D=C0H ?
 EC27     16 01                         LD      D,01H
 EC29     3A EBF7              PATGO1:  LD      A,(COLOR)
 EC2C     A2                            AND     D        ;D=40H,80H or 01H
 EC2D     C4 EC3D                       CALL    NZ,PATW
                               ;
 EC30     2A EBF5                       LD      HL,(PATB)
 EC33     01 4000                       LD      BC,4000H
 EC36     09                            ADD     HL,BC
 EC37     D8                            RET     C
 EC38     22 EBF5                       LD      (PATB),HL
 EC3B     18 DB                         JR      PATGO
                               ;...........................
 EC3D     7E                   PATW:    LD      A,(HL)
 EC3E     08                            EX      AF,AF'   ;A' HOLDS XCOUNT
 EC3F     23                            INC     HL
 EC40     7E                            LD      A,(HL)
 EC41     23                            INC     HL       ;HL=PAT-ADDR
 EC42     D9                            EXX
 EC43     57                            LD      D,A      ;D'=YCOUNT
 EC44     D9                            EXX
                               ;
 EC45     C5                   YLOPWB:  PUSH    BC       ;SAVE GPR-ADDR
 EC46     08                            EX      AF,AF'
 EC47     57                            LD      D,A      ;D=A'=XCOUNT
 EC48     08                            EX      AF,AF'   ;GET XCOUNT
                               ;
 EC49     ED 78                XLOPWB:  IN      A,(C)
 EC4B     AE                            XOR     (HL)
 EC4C     ED 79                         OUT     (C),A
 EC4E     23                            INC     HL
 EC4F     03                            INC     BC       ;OUT 1 BYTE
                               ;
```

```
 EC50     15                DEC    D
 EC51     C2 EC49           JP     NZ,XLOPWB        ;SAME LINE GO ON
                   ;DOWN 1 LINE
                   ;
 EC54     C1                POP    BC      ;BACK ADDR
 EC55     E5                PUSH   HL      ;SAVE P-ADDR
                   ;
 EC56     3E 08             LD     A,08H   ;VECTOR DOWN
 EC58     80                ADD    A,B     ;BC=ADDR, ADD HIGH BYTE
 EC59     47                LD     B,A
 EC5A     E6 C0             AND    0C0H    ;MASK 11000000
 EC5C     BB                CP     E       ;CHECK IN?
 EC5D     CA EC66           JP     Z,DONEWB         ;OK MASK
                   ;
 EC60     21 C050  PATCH1:  LD     HL,0C050H
                   ;OR       LD     HL,0C028H (WIDTH 40)
 EC63     09                ADD    HL,BC
 EC64     44                LD     B,H
 EC65     4D                LD     C,L
                   ;
 EC66     E1       DONEWB:  POP    HL
 EC67     D9                EXX
 EC68     15                DEC    D
 EC69     D9                EXX
 EC6A     C2 EC45           JP     NZ,YLOPWB
                   ;
 EC6D     C9                RET
                   ;
                   ;ADDR=4000H+(X1>>3)+((Y1 & 7)<<11) +(Y1>>3)*80
                   ;ADDR=4000H+(BC/8) +((L AND 7)<<11)+(HL/8) *80
                   ;HL=Y1,BC=X1,BREAKS HL,A,BC,DE
                   ;return HL=addr,A=mask,D=count
                   ;
 EC6E     7D       XYADDR:  LD     A,L     ;SAVE L
 EC6F     CD EC99           CALL   DIV8    ;HL=HL/8
                   ;
 EC72     54                LD     D,H
 EC73     5D                LD     E,L     ;DE=HL
                   ;
 EC74     29                ADD    HL,HL   ;
 EC75     29                ADD    HL,HL   ;80=16*5
 EC76     19                ADD    HL,DE   ;HL=5
 EC77     29                ADD    HL,HL   ;10
 EC78     29                ADD    HL,HL   ;20
 EC79     29                ADD    HL,HL   ;40
 EC7A     29       PATCH2:  ADD    HL,HL   ;80      ;HL=(HL/8)*80
                   ;OR       NOP                     (WIDTH 40)
                   ;
 EC7B     E6 07             AND    07H     ;A=(L AND 7)
 EC7D     87                ADD    A,A
 EC7E     87                ADD    A,A
 EC7F     87                ADD    A,A     ;A=((L AND 7)<<3)
                   ;
 EC80     C6 40             ADD    A,040H  ;ADD 4000H
 EC82     57                LD     D,A
 EC83     1E 00             LD     E,00H
                   ;DE=4000H+((L AND 7)<<(3+8))
 EC85     19                ADD    HL,DE   ;LAST 2 AND 1ST WERE DON
E
 EC86     EB                EX     DE,HL   ;HDE=HL (SAVE)
                   ;
 EC87     60                LD     H,B
 EC88     69                LD     L,C     ;HL=BC
                   ;
 EC89     7D                LD     A,L     ;FOR bit7-0(BC=X1)
 EC8A     CD EC99           CALL   DIV8    ;HL=HL/8
 EC8D     19                ADD    HL,DE   ;BADDR DONE
                   ;NOW HL= BEGIN ADDR
                   ;
 EC8E     E6 07             AND    07H     ;CALC bit7-0
 EC90     47                LD     B,A
 EC91     57                LD     D,A
 EC92     3E 80             LD     A,080H
                   ;
 EC94     C8                RET    Z       ;bit7 (<-AND 07H)

 EC95     0F       SFLP:    RRCA
 EC96     10 FD             DJNZ   SFLP
 EC98     C9                RET
                   ;
                   ;Now Acc has mask of 1 dot,
```

```
                                    ;HL has result (address),
                                    ;D has mask count (0-7).
                                    ;
 EC99     CB 3C                     DIV8:   SRL     H
 EC9B     CB 1D                             RR      L
 EC9D     CB 3C                             SRL     H
 EC9F     CB 1D                             RR      L
 ECA1     CB 3C                             SRL     H
 ECA3     CB 1D                             RR      L
 ECA5     C9                                RET
                                    ;
                                    ;DEFAULT MOUSE CURSOR
 ECA6     ECB6 ECD6                 ARW8:   DW      ARW80,ARW81 ;BIT SHIFTED PATTERN
S
 ECAA     ECF6 ED16                         DW      ARW82,ARW83
 ECAE     ED36 ED56                         DW      ARW84,ARW85
 ECB2     ED80 EDAA                         DW      ARW86,ARW87
                                    ;
 ECB6     03 0A                     ARW80:  DB      3,10
 ECB8     FF FC 00                          DB      0FFH,0FCH,000H
 ECBB     FF F0 00                          DB      0FFH,0F0H,000H
 ECBE     FF C0 00                          DB      0FFH,0C0H,000H
 ECC1     FF F0 00                          DB      0FFH,0F0H,000H
 ECC4     FF FC 00                          DB      0FFH,0FCH,000H
 ECC7     F3 FF 00                          DB      0F3H,0FFH,000H
 ECCA     C0 FF C0                          DB      0C0H,0FFH,0C0H
 ECCD     00 3F F0                          DB      000H,03FH,0F0H
 ECD0     00 0F C0                          DB      000H,00FH,0C0H
 ECD3     00 03 00                          DB      000H,003H,000H
                                    ;
 ECD6     03 0A                     ARW81:  DB      3,10
 ECD8     7F FE 00                          DB      07FH,0FEH,000H
 ECDB     7F F8 00                          DB      07FH,0F8H,000H
 ECDE     7F E0 00                          DB      07FH,0E0H,000H
 ECE1     7F F8 00                          DB      07FH,0F8H,000H
 ECE4     7F FE 00                          DB      07FH,0FEH,000H
 ECE7     79 FF 80                          DB      079H,0FFH,080H
 ECEA     60 7F E0                          DB      060H,07FH,0E0H
 ECED     00 1F F8                          DB      000H,01FH,0F8H
 ECF0     00 07 E0                          DB      000H,007H,0E0H
 ECF3     00 01 80                          DB      000H,001H,080H
                                    ;
 ECF6     03 0A                     ARW82:  DB      3,10
 ECF8     3F FF 00                          DB      03FH,0FFH,000H
 ECFB     3F FC 00                          DB      03FH,0FCH,000H
 ECFE     3F F0 00                          DB      03FH,0F0H,000H
 ED01     3F FC 00                          DB      03FH,0FCH,000H
 ED04     3F FF 00                          DB      03FH,0FFH,000H
 ED07     3C FF C0                          DB      03CH,0FFH,0C0H
 ED0A     30 3F F0                          DB      030H,03FH,0F0H
 ED0D     00 0F FC                          DB      000H,00FH,0FCH
 ED10     00 03 F0                          DB      000H,003H,0F0H
 ED13     00 00 C0                          DB      000H,000H,0C0H
                                    ;
 ED16     03 0A                     ARW83:  DB      3,10
 ED18     1F FF 80                          DB      01FH,0FFH,080H
 ED1B     1F FE 00                          DB      01FH,0FEH,000H
 ED1E     1F F8 00                          DB      01FH,0F8H,000H
 ED21     1F FE 00                          DB      01FH,0FEH,000H
 ED24     1F FF 80                          DB      01FH,0FFH,080H
 ED27     1E 7F E0                          DB      01EH,07FH,0E0H
 ED2A     18 1F F8                          DB      018H,01FH,0F8H
 ED2D     00 07 FE                          DB      000H,007H,0FEH
 ED30     00 01 F8                          DB      000H,001H,0F8H
 ED33     00 00 60                          DB      000H,000H,060H
                                    ;
 ED36     03 0A                     ARW84:  DB      3,10
 ED38     0F FF C0                          DB      00FH,0FFH,0C0H
 ED3B     0F FF 00                          DB      00FH,0FFH,000H
 ED3E     0F FC 00                          DB      00FH,0FCH,000H
 ED41     0F FF 00                          DB      00FH,0FFH,000H
 ED44     0F FF C0                          DB      00FH,0FFH,0C0H
 ED47     0F 3F F0                          DB      00FH,03FH,0F0H
 ED4A     0C 0F FC                          DB      00CH,00FH,0FCH
 ED4D     00 03 FF                          DB      000H,003H,0FFH
 ED50     00 00 FC                          DB      000H,000H,0FCH
 ED53     00 00 30                          DB      000H,000H,030H
                                    ;
 ED56     04 0A                     ARW85:  DB      4,10
 ED58     07 FF E0 00                       DB      007H,0FFH,0E0H,000H
 ED5C     07 FF 80 00                       DB      007H,0FFH,080H,000H
```

```
ED60    07 FE 00 00              DB      007H,0FEH,000H,000H
ED64    07 FF 80 00              DB      007H,0FFH,080H,000H
ED68    07 FF E0 00              DB      007H,0FFH,0E0H,000H
ED6C    07 9F F8 00              DB      007H,09FH,0F8H,000H
ED70    06 07 FE 00              DB      006H,007H,0FEH,000H
ED74    00 01 FF 80              DB      000H,001H,0FFH,080H
ED78    00 00 7E 00              DB      000H,000H,07EH,000H
ED7C    00 00 18 00              DB      000H,000H,018H,000H
                         ;
ED80    04 0A            ARW86:  DB      4,10
ED82    03 FF F0 00              DB      003H,0FFH,0F0H,000H
ED86    03 FF C0 00              DB      003H,0FFH,0C0H,000H
ED8A    03 FF 00 00              DB      003H,0FFH,000H,000H
ED8E    03 FF C0 00              DB      003H,0FFH,0C0H,000H
ED92    03 FF F0 00              DB      003H,0FFH,0F0H,000H
ED96    03 CF FC 00              DB      003H,0CFH,0FCH,000H
ED9A    03 03 FF 00              DB      003H,003H,0FFH,000H
ED9E    00 00 FF C0              DB      000H,000H,0FFH,0C0H
EDA2    00 00 3F 00              DB      000H,000H,03FH,000H
EDA6    00 00 0C 00              DB      000H,000H,00CH,000H
                         ;
EDAA    04 0A            ARW87:  DB      4,10
EDAC    01 FF F8 00              DB      001H,0FFH,0F8H,000H
EDB0    01 FF E0 00              DB      001H,0FFH,0E0H,000H
EDB4    01 FF 80 00              DB      001H,0FFH,080H,000H
EDB8    01 FF E0 00              DB      001H,0FFH,0E0H,000H
EDBC    01 FF F8 00              DB      001H,0FFH,0F8H,000H
EDC0    01 E7 FE 00              DB      001H,0E7H,0FEH,000H
EDC4    01 81 FF 80              DB      001H,081H,0FFH,080H
EDC8    00 00 7F E0              DB      000H,000H,07FH,0E0H
EDCC    00 00 1F 80              DB      000H,000H,01FH,080H
EDD0    00 00 06 00              DB      000H,000H,006H,000H
                         ;
EDD4    EDE4 EDFA        ARW4:   DW      ARW40,ARW41 ;BIT SHIFTED PATTERN
S
EDD8    EE10 EE26                DW      ARW42,ARW43
EDDC    EE3C EE52                DW      ARW44,ARW45
EDE0    EE68 EE7E                DW      ARW46,ARW47
                         ;
EDE4    02 0A            ARW40:  DB      2,10
EDE6    FE 00                    DB      0FEH,000H
EDE8    FC 00                    DB      0FCH,000H
EDEA    F8 00                    DB      0F8H,000H
EDEC    FC 00                    DB      0FCH,000H
EDEE    FE 00                    DB      0FEH,000H
EDF0    DF 00                    DB      0DFH,000H
EDF2    8F 80                    DB      08FH,080H
EDF4    07 C0                    DB      007H,0C0H
EDF6    03 80                    DB      003H,080H
EDF8    01 00                    DB      001H,000H
                         ;
EDFA    02 0A            ARW41:  DB      2,10
EDFC    7F 00                    DB      07FH,000H
EDFE    7E 00                    DB      07EH,000H
EE00    7C 00                    DB      07CH,000H
EE02    7E 00                    DB      07EH,000H
EE04    7F 00                    DB      07FH,000H
EE06    6F 80                    DB      06FH,080H
EE08    47 C0                    DB      047H,0C0H
EE0A    03 E0                    DB      003H,0E0H
EE0C    01 C0                    DB      001H,0C0H
EE0E    00 80                    DB      000H,080H
                         ;
EE10    02 0A            ARW42:  DB      2,10
EE12    3F 80                    DB      03FH,080H
EE14    3F 00                    DB      03FH,000H
EE16    3E 00                    DB      03EH,000H
EE18    3F 00                    DB      03FH,000H
EE1A    3F 80                    DB      03FH,080H
EE1C    37 C0                    DB      037H,0C0H
EE1E    23 E0                    DB      023H,0E0H
EE20    01 F0                    DB      001H,0F0H
EE22    00 E0                    DB      000H,0E0H
EE24    00 40                    DB      000H,040H
                         ;
EE26    02 0A            ARW43:  DB      2,10
EE28    1F C0                    DB      01FH,0C0H
EE2A    1F 80                    DB      01FH,080H
EE2C    1F 00                    DB      01FH,000H
EE2E    1F 80                    DB      01FH,080H
EE30    1F C0                    DB      01FH,0C0H
```

```
EE32    1B E0                 DB     01BH,0E0H
EE34    11 F0                 DB     011H,0F0H
EE36    00 F8                 DB     000H,0F8H
EE38    00 70                 DB     000H,070H
EE3A    00 20                 DB     000H,020H
                      ;
EE3C    02 0A         ARW44:  DB     2,10
EE3E    0F E0                 DB     00FH,0E0H
EE40    0F C0                 DB     00FH,0C0H
EE42    0F 80                 DB     00FH,080H
EE44    0F C0                 DB     00FH,0C0H
EE46    0F E0                 DB     00FH,0E0H
EE48    0D F0                 DB     00DH,0F0H
EE4A    08 F8                 DB     008H,0F8H
EE4C    00 7C                 DB     000H,07CH
EE4E    00 38                 DB     000H,038H
EE50    00 10                 DB     000H,010H
                      ;
EE52    02 0A         ARW45:  DB     2,10
EE54    07 F0                 DB     007H,0F0H
EE56    07 E0                 DB     007H,0E0H
EE58    07 C0                 DB     007H,0C0H
EE5A    07 E0                 DB     007H,0E0H
EE5C    07 F0                 DB     007H,0F0H
EE5E    06 F8                 DB     006H,0F8H
EE60    04 7C                 DB     004H,07CH
EE62    00 3E                 DB     000H,03EH
EE64    00 1C                 DB     000H,01CH
EE66    00 08                 DB     000H,008H
                      ;
EE68    02 0A         ARW46:  DB     2,10
EE6A    03 F8                 DB     003H,0F8H
EE6C    03 F0                 DB     003H,0F0H
EE6E    03 E0                 DB     003H,0E0H
EE70    03 F0                 DB     003H,0F0H
EE72    03 F8                 DB     003H,0F8H
EE74    03 7C                 DB     003H,07CH
EE76    02 3E                 DB     002H,03EH
EE78    00 1F                 DB     000H,01FH
EE7A    00 0E                 DB     000H,00EH
EE7C    00 04                 DB     000H,004H
                      ;
EE7E    03 0A         ARW47:  DB     3,10
EE80    01 FC 00              DB     001H,0FCH,000H
EE83    01 F8 00              DB     001H,0F8H,000H
EE86    01 F0 00              DB     001H,0F0H,000H
EE89    01 F8 00              DB     001H,0F8H,000H
EE8C    01 FC 00              DB     001H,0FCH,000H
EE8F    01 BE 00              DB     001H,0BEH,000H
EE92    01 1F 00              DB     001H,01FH,000H
EE95    00 0F 80              DB     000H,00FH,080H
EE98    00 07 00              DB     000H,007H,000H
EE9B    00 02 00              DB     000H,002H,000H
                      ;
                              END
```

リスト 6-3　マウスドライバ(リスト 6-2)のダンプリスト，もしくは
"MOUSE.OBJ"でセーブしてねとあの子が言った

```
EA00 EB 5E 23 56 EB 22 29 EB : E3
EA08 11 31 EB 01 18 00 ED B0 : E3
EA10 3E 50 32 61 EC 3E 29 32 : A6
EA18 7A EC 3E 28 32 AC EA 21 : B5
EA20 A6 EC 22 2F EB 3A 48 EB : 3B
EA28 0F 38 17 47 3E 28 32 61 : 9E
EA30 EC 3E 00 32 7A EC 3E 18 : 18
EA38 32 AC EA 21 D4 ED 22 2F : FB
EA40 EB 78 0F 30 04 CD D9 EB : 37
EA48 C9 2A 31 EB 22 2B EB 2A : 71
EA50 33 EB 22 2D EB CD DC EA : EB
EA58 E6 03 28 05 3E 02 C3 16 : 2F
EA60 EB CD A5 EB CD D9 EB FD : D6
EA68 21 39 EB CD 49 EB 3A 39 : B9
EA70 EB F5 FD 21 39 EB CD 49 : 38
```

```
EA78 EB F1 B7 20 11 21 35 EB : 05
---------------------------------
SUM: 36 55 6F EF 47 DE 8D 00 1738

EA80 3A 39 EB E6 03 06 04 BE : 0F
EA88 23 CA 11 EB 10 F9 21 3A : 4D
EA90 EB 7E 23 B6 28 D8 3A 3A : B6
EA98 EB CD 0A EB 2A 31 EB 19 : 0C
EAA0 22 2B EB 3A 3B EB CD 0A : 6F
EAA8 EB 3E 01 BB 28 04 CB 2A : 06
EAB0 CB 1B 2A 33 EB 19 22 2D : 96
EAB8 EB CD DC EA FE 03 28 AE : 55
EAC0 F5 CD D9 EB F1 0F 38 06 : C4
EAC8 2A 2B EB 22 31 EB 0F 38 : C5
EAD0 06 2A 2D EB 22 33 EB CD : 55
```

```
EAD8  D9  EB  18  92  AF  ED  5B  44  :  A9
EAE0  EB  2A  2D  EB  B7  ED  52  38  :  5B
EAE8  09  ED  5B  2D  EB  2A  46  EB  :  C4
EAF0  ED  52  17  ED  5B  40  EB  2A  :  F3
EAF8  2B  EB  B7  ED  52  38  09  ED  :  3A
------------------------------------
SUM:  00  00  7A  00  F3  BC  45  E3  B461

EB00  5B  2B  EB  2A  42  EB  ED  52  :  07
EB08  17  C9  5F  16  00  07  D0  15  :  41
EB10  C9  AF  18  02  3E  01  32  3C  :  3F
EB18  EB  CD  D9  EB  ED  5B  29  EB  :  D8
EB20  21  31  EB  01  18  00  ED  B0  :  F3
EB28  C9  1F  37  CD  FC  0C  E6  A3  :  7D
EB30  47  E6  80  78  CA  32  1F  32  :  72
EB38  BE  3E  06  80  22  BF  3E  E6  :  87
EB40  10  C4  A9  04  3A  EA  3C  87  :  68
EB48  3E  01  93  1F  3E  05  ED  79  :  9A
EB50  AF  ED  79  16  80  15  20  FD  :  DD
EB58  3E  05  ED  79  3E  02  ED  79  :  4F
EB60  1E  05  FD  22  80  EB  16  03  :  C6
EB68  CD  82  EB  38  09  FD  77  00  :  EF
EB70  FD  23  15  20  F3  C9  FD  2A  :  38
EB78  80  EB  1D  20  E9  C3  14  EB  :  53
------------------------------------
SUM:  B8  30  9F  3F  08  C5  1C  87  0616

EB80  1C  E5  01  93  1F  21  00  00  :  D5
EB88  2B  7C  B5  28  0D  AF  ED  79  :  A6
EB90  ED  78  1F  30  F3  0B  ED  78  :  17
EB98  B7  C9  D5  CD  A5  EB  D1  2B  :  AE
EBA0  7C  B5  20  FB  C9  F3  CD  AE  :  83
EBA8  EB  CD  BA  EB  FB  C9  01  A2  :  C4
EBB0  1F  3E  47  ED  79  3E  1A  ED  :  4F
EBB8  79  C9  01  93  1F  21  CA  EB  :  CB
EBC0  16  0F  7E  23  ED  79  15  20  :  61
EBC8  F9  C9  18  01  00  02  70  04  :  51
EBD0  44  05  00  06  00  07  00  03  :  59
EBD8  C1  ED  4B  31  EB  2A  33  EB  :  5D
EBE0  ED  5B  3D  EB  7A  B3  20  04  :  C1
EBE8  ED  5B  2F  EB  3A  3F  EB  CD  :  93
EBF0  F8  EB  C9  3D  FE  58  C2  00  :  01
EBF8  ED  53  F3  EB  0F  0F  32  F7  :  65
------------------------------------
SUM:  BD  E9  D5  77  B9  E6  14  1E  3C14

EC00  EB  CD  6E  EC  22  F5  EB  6A  :  7E
EC08  26  00  29  ED  5B  F3  EB  19  :  8E
EC10  11  F3  EB  01  02  00  ED  B0  :  8F
EC18  2A  F3  EB  ED  4B  F5  EB  78  :  98
EC20  E6  C0  5F  57  E2  29  EC  16  :  69
EC28  01  3A  F7  EB  A2  C4  3D  EC  :  AC
EC30  2A  F5  EB  01  00  40  09  D8  :  2C
EC38  22  F5  EB  18  DB  7E  08  23  :  9E
EC40  7E  23  D9  57  D9  C5  08  57  :  CE
EC48  08  ED  78  AE  ED  79  23  03  :  A7
EC50  15  C2  49  EC  C1  E5  3E  08  :  F8
EC58  80  47  E6  C0  BB  CA  66  EC  :  44
EC60  21  50  C0  09  44  4D  E1  D9  :  85
EC68  15  D9  C2  45  EC  C9  7D  CD  :  F4
EC70  99  EC  54  5D  29  29  19  29  :  CA
EC78  29  29  29  E6  07  87  87  87  :  FD
------------------------------------
SUM:  92  EE  18  64  CB  3B  B5  4C  15F3

EC80  C6  40  57  1E  00  19  EB  60  :  DF
EC88  69  7D  CD  99  EC  19  E6  07  :  3E
EC90  47  57  3E  80  C8  0F  10  FD  :  40
EC98  C9  CB  3C  CB  1D  CB  3C  CB  :  8A
ECA0  1D  CB  3C  CB  1D  C9  B6  EC  :  77
ECA8  D6  EC  F6  EC  16  ED  36  ED  :  CA
ECB0  56  ED  80  ED  AA  ED  03  0A  :  54
ECB8  FF  FC  00  FF  F0  00  FF  C0  :  A9
ECC0  00  FF  F0  00  FF  FC  00  F3  :  DD
ECC8  FF  00  C0  FF  C0  00  3F  F0  :  AD
ECD0  00  0F  C0  00  03  00  03  0A  :  DF
ECD8  7F  FE  00  7F  F8  00  7F  E0  :  53
ECE0  00  7F  F8  00  7F  FE  00  79  :  6D
ECE8  FF  80  60  7F  E0  00  1F  F8  :  55
ECF0  00  07  E0  00  01  80  03  0A  :  75
ECF8  3F  FF  00  3F  FC  00  3F  F0  :  A8
------------------------------------
SUM:  43  90  F8  E1  B4  29  2D  0A  FF0E

ED00  00  3F  FC  00  3F  FF  00  3C  :  B5
ED08  FF  C0  30  3F  F0  00  0F  FC  :  29
ED10  00  03  F0  00  00  C0  03  0A  :  C0
ED18  1F  FF  80  1F  FE  00  1F  F8  :  D2
ED20  00  1F  FE  00  1F  FF  80  1E  :  D9
ED28  7F  E0  18  1F  F8  00  07  FE  :  93
ED30  00  01  F8  00  00  60  03  0A  :  66
ED38  0F  FF  C0  0F  FF  00  0F  FC  :  E7
ED40  00  0F  FF  00  0F  FF  C0  0F  :  EB
ED48  3F  F0  0C  0F  FC  00  03  FF  :  48
ED50  00  00  FC  00  00  30  04  0A  :  3A
ED58  07  FF  E0  00  07  FF  80  00  :  6C
ED60  07  FE  00  00  07  FF  80  00  :  8B
ED68  07  FF  E0  00  07  9F  F8  00  :  84
ED70  06  07  FE  00  00  01  FF  80  :  8B
ED78  00  00  7E  00  00  00  18  00  :  96
------------------------------------
SUM:  06  02  AD  9B  63  EB  A0  F4  A45C

ED80  04  0A  03  FF  F0  00  03  FF  :  02
ED88  C0  00  03  FF  00  00  03  FF  :  C4
ED90  C0  00  03  FF  F0  00  03  CF  :  84
ED98  FC  00  03  03  FF  00  00  00  :  01
EDA0  FF  C0  00  00  3F  00  00  00  :  FE
EDA8  0C  00  04  0A  01  FF  F8  00  :  12
EDB0  01  FF  E0  00  01  FF  80  00  :  60
EDB8  01  FF  E0  00  01  FF  F8  00  :  D8
EDC0  01  E7  FE  00  01  81  FF  80  :  E7
EDC8  00  00  7F  E0  00  00  1F  80  :  FE
EDD0  00  00  06  00  E4  ED  FA  ED  :  BE
EDD8  10  EE  26  EE  3C  EE  52  EE  :  7C
EDE0  68  EE  7E  EE  02  0A  FE  00  :  CC
EDE8  FC  00  F8  00  FC  00  FE  00  :  EE
EDF0  DF  00  8F  80  07  C0  03  80  :  38
EDF8  01  00  02  0A  7F  00  7E  00  :  0A
------------------------------------
SUM:  E2  8B  80  50  C6  23  60  28  E48B

EE00  7C  00  7E  00  7F  00  6F  80  :  68
EE08  47  C0  03  E0  01  C0  00  80  :  2B
EE10  02  0A  3F  80  3F  00  3E  00  :  48
EE18  3F  00  3F  80  37  C0  23  E0  :  F8
EE20  01  F0  00  E0  00  40  02  0A  :  1D
EE28  1F  C0  1F  80  1F  00  1F  80  :  3C
EE30  1F  C0  1B  E0  11  F0  00  F8  :  D3
EE38  00  70  00  20  02  0A  0F  E0  :  8B
EE40  0F  C0  0F  80  0F  C0  0F  E0  :  1C
EE48  0D  F0  08  F8  00  7C  00  38  :  B1
EE50  00  10  02  0A  07  F0  07  E0  :  FA
EE58  07  C0  07  E0  07  F0  06  F8  :  A3
EE60  04  7C  00  3E  00  1C  00  08  :  E2
EE68  02  0A  03  F8  03  F0  03  E0  :  DD
EE70  03  F0  03  F8  03  7C  02  3E  :  AD
EE78  00  1F  00  0E  00  04  03  0A  :  3E
```

```
---------------------------------
SUM: 6F BF 5F DE 4B 62 24 62 B9D0

EE80 01 FC 00 01 F8 00 01 F0 : E7
EE88 00 01 F8 00 01 FC 00 01 : F7
```

```
EE90 BE 00 01 1F 00 00 0F 80 : 6D
EE98 00 07 00 00 02 00       : 09
---------------------------------
SUM: BF 04 F9 20 FB FC 10 71 39E5
```

**リスト 6-4　マウスが舞うす，もしくは早くパターンエディタになりたい**

```
100 CLEAR &HE000
110 IF MEM$(&HEA00,4)<>HEXCHR$("EB 5E 23 56") THEN LOADM "MOUSE.OBJ"
120 DEFUSR0=&HEA00
130 X=100:Y=100
140 MEM$(&HE000,4)=MKI$(X)+MKI$(Y)                         :'X Y
150 'MEM$(&HE000,4)=HEXCHR$("64 00 64 00")                  :'X Y
160 MEM$(&HE004,4)=HEXCHR$("01 02 03 04")                  :'BUTTON CONDITION
170 MEM$(&HE008,4)=HEXCHR$("00 00 00 00")                  :'MOUSE DATA
180 MEM$(&HE00C,3)=HEXCHR$("00 00 04")                     :'MOUSE CURSOR
190 X0=40:X1=240:Y0=24:Y1=160
200 MEM$(&HE00F,8)=MKI$(X0)+MKI$(X1-1)+MKI$(Y0)+MKI$(Y1-1)
210 'MEM$(&HE00F,8)=HEXCHR$("28 00 EF 00 18 00 9F 00"):'AREA
220 MEM$(&HE017,1)=HEXCHR$("00")                           :'MODE
230 WIDTH40
240 '
250 'MEM$(&HE00C,3) =HEXCHR$("80 D0 07"):'ﾄ ｽﾙﾄ CURSOR ｶﾞ '.' ﾆ ﾅﾙ
260 'MEM$(&HE080,16)=HEXCHR$("90 D0 93 D0 96 D0 99 D0 9C D0 9F D0 A2 D0 A5 D0")
270 'MEM$(&HE090,16)=HEXCHR$("01 01 80 01 01 40 01 01 20 01 01 10 01 01 08 01")
280 'MEM$(&HE0A0,16)=HEXCHR$("01 04 01 01 02 01 01 01")
290 CLS 4:INIT
300 LINE(X0-1,Y0-1)-(X1,Y1),PSET,7,B
310 LINE((X0-1)/8,(Y0-1)/8)-((X1-8)/8,(Y1-8)/8),CHR$(225),BF
320 '
330 D$=USR0(CHR$(&H0,&HE0))
340 '
350 BT=PEEK(&HE008) AND 3
360 IF BT=3 THEN WIDTH80:END
370 IF BT=1 THEN C=224:GOSUB "PSET"
380 IF BT=2 THEN C=225:GOSUB "PSET"
390 GOTO 330
400 '
410 LABEL"PSET"
420 X=CVI(MEM$(&HE000,2)):X=INT(X/8)
430 Y=CVI(MEM$(&HE002,2)):Y=INT(Y/8)
440 PSET(X+239+1,Y+159+1,(225-C)*7)
450 LOCATEX,Y:PRINTCHR$(C);:RETURN
```

打ち込んだなら，

SAVEM "MOUSE. OBJ", &HEA00, &HEE9D

でセーブしていただきたい（テープでも可だけど，はっきり言ってマウスより先にディスクを買うべきである）。次にリスト 6-4 を RUN する（おっと，turbo BASIC の場合は事前に「KMODE 0」を実行しておくこと）。するとディスクが回り，"MOUSE. OBJ" がロードされた後，何やらパターンエディタもどきの画面になるであろう。セコイことにモノクロで，**気分の良いことに**セーブ機能などは一切付いていない。なぜならばこのプログラムは，私が苦心してデザインしたマウスカーソル（本当はただの左上向きの矢印だけど）が，画面を華麗に舞う姿を楽しむためだけ，のものだからである。左側のボタンを押すと白丸が，右側のボタンを押すと中抜き丸が表示されるであろう。終わりたいときは両方のボタンを**同時に**押すのであるが，人間の指はどうしても左右のボタンを押すのに時間差を生じてしまうらしく，なかなかに難しいはずである。いざとなったら，[SHIFT]+[BREAK] をちょんと押した後，どちらか片方のボタンで BREAK する。

まずはリスト 6-4 の説明，すなわちリスト 6-3 の機械語ルーチンの使い方である。140〜220 行はパラメータの指定である。中身は，

**① マウスカーソルの位置（X，Y の順）**

2 バイトの値で指定する。MKI$の使い方に注意。合計 4 バイトである。

**② どのボタンが押されたらリターンするか**

これは 4 とおり指定できる。

$00_H$＝ボタンが押されてなかったら RET

$01_H$＝左側ボタンだけが押されたら RET

$02_H$＝右側ボタンだけが押されたら RET

$03_H$＝両側とも押されたら RET

$04_H$＝無効

リスト 6-4 では，とにかく何かボタンが押されたら RET するように $01_H$，$02_H$，$03_H$の 3 とおりを指定している。$04_H$は「詰め物」のよーなものである。合計 4 バイト。

**③ マウスデータ**

最初の 3 バイトには，最後にマウスから読み出された生のデータが入っている。 4 バイト目はエラーコードである。

$00_H$→正常

$01_H$→マウスがつながっていないなど

$02_H$→マウスカーソルの位置が，最初からウィンドウの外側にある。

合計 4 バイト。

**④ マウスカーソルの形状，色の指定**

最初の 2 バイトはカーソル用のグラフィックのデータが格納されているアドレスである。$0000_H$を指定すると，デフォルトの左上向き矢印になる。3 バイト目は色の指定。リスト 6-4 では緑になっている。合計 3 バイト。

**⑤ ウィンドウ（移動範囲）の指定**

X 方向の下限値，X 方向の上限値，Y 方向の下限値，Y 方向の上限値，の順に指定する。それぞれ 2 バイトの値である。リスト 5 では，マウスカーソルの左上端の位置が，(40，24)－(240－1，160－1）の囲いの中から出ないようにしている。合計 8 バイト。

**⑥ モードの指定**

第 0 ビットはWIDTHの指定。 0 なら40字モード， 1 なら80字モードとなる。第 1 ビットはマウス動作をするか，ただのグラフィック描画をするかのフラグである。 0 ならマウス動作，1 なら④で指定されたマウスカーソルを XOR モードで表示するだけですぐ RET する。合計 1 バイト。

ま，大体以上なわけである。よって，正しいリスト 6-3 の使い方は，①～⑥のパラメータを指定して，330 行のようにそのパラメータの先頭番地をもって，USR 命令である。その後，ボタンの状態を知りたいのなら③の先頭 1 バイト（リスト 6-4 では $E008_H$番地）を PEEK して AND 3，マウスカーソルの位置を知りたいなら，420 行，430 行のようにすればよい。その後 8 で割っているのはキャラクタの位置に変換しているのだから勘違いはしないように。

それではたちまちリスト 6-2 の説明に入る。最初の $EA0F_H$まではパラメータの転送など

である。扱いやすいようにEB31$_H$からの24バイトに転送して使うのである（RETするときには，元に転送し直す）。

次からEA41$_H$までは，40/80字モードの切り換えで，プログラムを直接書き換えるという最低のテクニックを使っている。パラメータの⑥の第1ビットが1ならば，EA45$_H$番地に来てたちまちRET，さもなくばEA49$_H$からのマウスドライバに入る。

後は詳しい説明はしないで，サブルーチンの説明にとどめておく。

CIN＝EADC$_H$は，（X0，Y0）がウィンドウの中に入っているかどうかの判定ルーチン。ウィンドウの中に入っていなければAレジスタのビットを立てる（ビット1はY，ビット0はX）。

XSIGN＝EB0A$_H$は8ビットの符号付きのEレジスタを16ビット値にしてDEレジスタに入れるもの。

MSIN　＝EB49$_H$はマウスからの入力

SETMS　＝EBA5$_H$はマウスのイニシャライズ

PAT　＝EBF8$_H$はマウスカーソルの描画

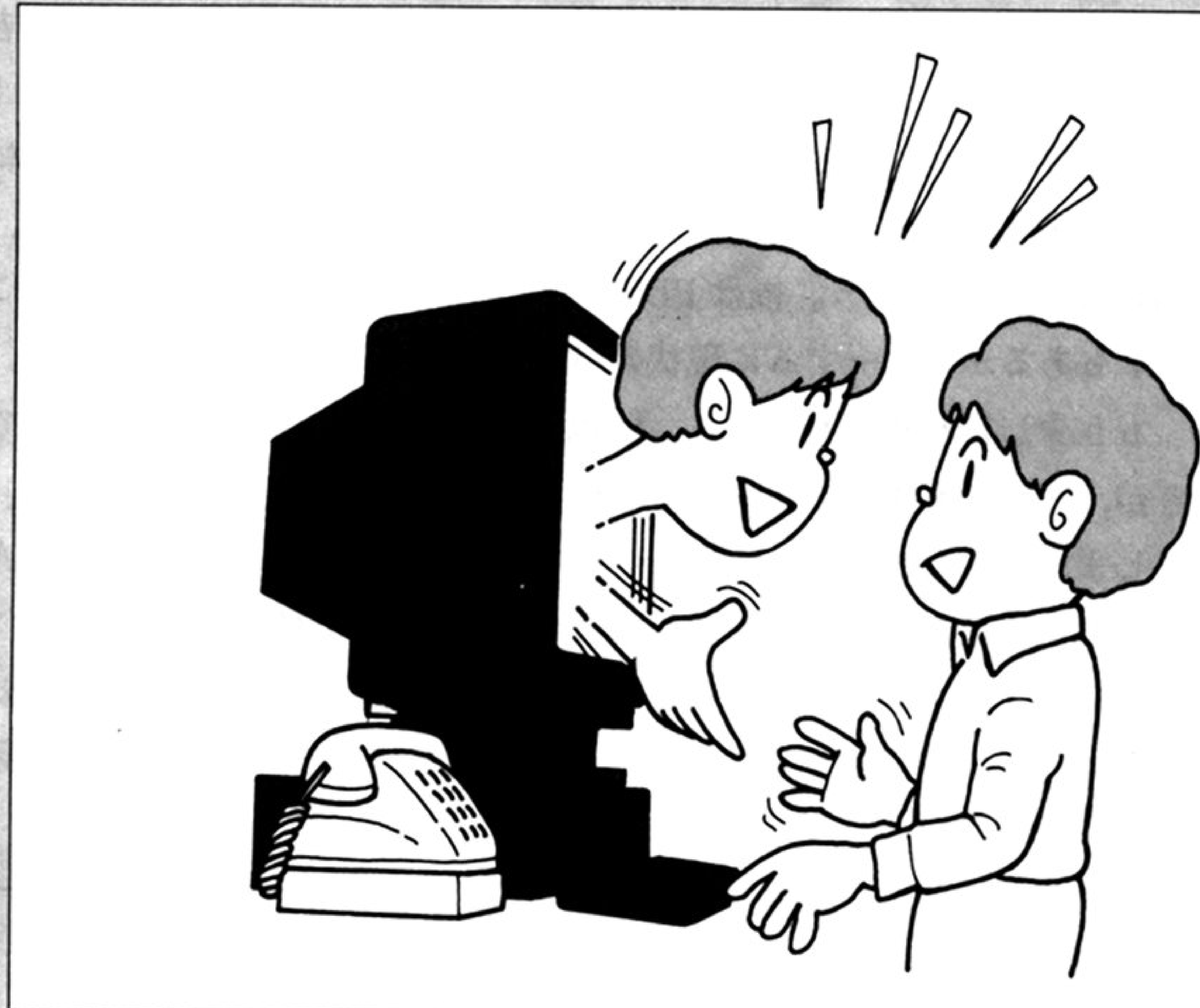
XYADDR ＝EC6E$_H$は座標(X，Y)からG-RAMのアドレスを計算するルーチン

そして，ECA6$_H$以降はグラフィックデータである。グラフィックデータは描画をスムースにするため，あらかじめ1ビットずつシフトしたデータを8とおり用意しておく方法を採っている。メモリをけちって，実行時にいちいちシフトさせるとマウスの操作性がガクンと落ちるのである。40字モードと80字モードの2とおりのデータで500バイト近くある。清く正しいプログラムでは，起動直前にシフトしたデータを作るべきなのであるが，**しっかり手抜きを**しているわけである。

では，「マウスを生かすも殺すもソフトしだい」とつぶやきつつ，この章を終わるのである。

# 第7章

# 通信



通信だってするのである

# 第7章
# 通信だってするのである・・・・・・・

この章では前章の舌の根も乾かぬうちに通信をするのである。具体的には CZ-8BM2（RS-232C・マウスボード）用のコントロールプログラムなのである。プログラムは turbo BASIC の RS-232C 関係の命令と同等のことができるようにすることを目的として作った。よって turbo ユーザーにとっては，基本的にお呼びでない。しかしプログラム中の SIO, CTC のアドレスをずらし，さらに CZ-8FB01 を使うか（もしくは CZ-8FB02 であっても割り込みベクトルのつじつまを合わせる）すれば turbo でも使えることは使えるから，「BASIC に依存しない通信プログラム」を作る際の参考ぐらいにはなるであろう。

で，実は CZ-8BM2 にはカセットに入ったアプリケーションプログラム（機械語）が付いてくるのであるが，どうやらこの章のプログラムはそれよりはましなものになったようである。

ところで通信であるが，私の持論では**「通信＝不毛」**ということになっている。誰が何と言っても現時点では（1987 年である）NTT が儲かるだけなのだ。モデムに必要なモジュラージャックを取り付ける工事だけで何千円もボるそうな（資格のない人が取り付けるのは禁止されている）。また，日本ではぼちぼちと通信速度が 1200 bps，さらにはそれ以上へと移行しつつあるようだが，はっきり言ってそれでもまだタコなのである。そして一番の問題は「パソコン通信なんかやってて何が面白いの？」である。おそらく面白いのは CHAT だけであろう。後は PDS（パブリックドメインソフト）が手に入るなどというメリットもあるが，よーするに現状のパソコン通信は「それだけのもの」なのである。

もう少し言わせてもらうと，**私は CHAT が嫌いである**。やってみれば楽しいのだろうが，それでも嫌いである。なぜかと言うと，文明社会の「ユガミ」みたいなものを感じてしまうからなのだ。これはあくまでも想像するだけなのだが，CHAT の情熱の嵐に巻き込まれたら最後なのではないだろうか。すなわち CHAT に命をかけているよーな人は，他人とのコミュニケーションにどっかこっか問題があるのではないかと思えるのである。面と向かって肉声で話すよりも，キーボードをパコパコ叩いて文字で話す方がいいなんて，なんか不自然だろう。

というわけで，この章のプログラムは，できればメディアコンバートなどに使い，**くれぐれも**パソコン通信などには使わないよーにお願いしたい。

## それではいきなり始める

X1 のユーザーは turbo BASIC のマニュアルを持っていないわけだから，**表 7-1**（turboZ のマニュアルから引用）を参考にしていただきたい。ただしプログラムは BASIC のコ

マンドを拡張するわけではないので，USR0～USR9を使うことになっている。その際，文字列変数を渡すときには「その長さ」にちょいと気を付けなければいけないので，サンプルプログラムをよく見て使うように。なお，SAVE, SAVEM, LOAD, LOADMはないので，自分で(BASICなどで)プログラムを組むなどして解決するよーに。

表7-1　通信パラメータの説明

通信パラメータには次のものがある。
1）ボーレート
2）パリティ
3）データビット長
4）ストップビット長
5）通信制御指定
6）カナの表現方法指定
7）CR，LFコードの送信処理
8）CR，LFコードの受信処理
9）日本語文字列の表現方法指定（本書のプログラムでは"N"しか使えない）
10）エンドコード指定

**1）ボーレート**
0～6の数値によりボーレートを指定する。

| 数値 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| ボーレート（bps） | 150 | 300 | 600 | 1200 | 2400 | 4800 | 9600 |

**2）パリティ**
E：偶数パリティチェックを使用する。
O：奇数パリティチェックを使用する。
N：パリティチェックを使用しない。

**3）データビット長**
5：データビット長を5ビットとする。
6：データビット長を6ビットとする。
7：データビット長を7ビットとする。
8：データビット長を8ビットとする。

**4）ストップビット長**
1：ストップビット長を1ビットとする。
2：ストップビット長を1.5ビットとする。
3：ストップビット長を2ビットとする。

**5）通信制御指定**
X：XON/XOFFコードによる制御を行なう。
R：RTS制御信号のON/OFFによる制御をする。
Nまたは省略：通信制御を行なわない。
この通信制御とは，データ送受信時に受信側においてデータの受信処理が間に合わないとき，送信側に対して送信の一時停止を要求し受信できる状態になると送信再開を要求する方法のことをいう。
"X"を指定すると，データとしてXOFFコード(&H13)を送信側へ送ることによって送信の一時停止を要求し，XONコード(&H11)により送信再開を要求する。
"R"を指定すると，RTS信号線をOFFにすることによって送信の一時停止を要求し，ONにすることにより送信再開を要求する。

**6）カナの表現方法指定**
S：データビット長7ビットでカナの送受信ができる。
Nまたは省略：データビット長7ビットでカナの送受信ができない。
データビット長7ビットでのカナの送受信はシフトインコードSI(&H0F)があると，それ以降のデータをカナとして処理し，シフトアウトコードSO(&H0E)があればそれ以降のデータを英数字として処理する。

**7）CR，LFコードの送信処理**
C：復帰改行コードとしてCRコード（&H0D）を送信する。
Lまたは省略：復帰改行コードとしてCRコード(&H0D)＋LFコード(&H0A)を送信する。

**8）CR，LFコードの受信処理**
C：CRコード(&H0D)の受信によって復帰＋改行処理する。
Lまたは省略：CRコード(&H0D)のみを受信するとそれはデータとして処理される。CRコード(&H0D)＋LFコード（&H0A）を連続して受信すると，復帰＋改行処理をする。

**9）日本語文字列の表現方法指定**
J：JIS漢字コードを使用する。漢字インコードKI(&H1B4B)で日本語文字列の始まりを示し，漢字アウトコードKO(&H1B48)で日本語文字列の終わりを示す。
Nまたは省略：シフトJIS漢字コードを使用する。(本書のプログラムでは"N"しか使えない）

**10）エンドコード指定**
データ転送の終了を判断するためのエンドコードを指定する。エンドコードとしてコントロールコード(&H00～&H1F)の中から任意の一つを選択できる。実際に指定するパラメータは，キャラクターコード &H40～&H5F に対応する文字を使用する（&H60～&H7Fも可)。たとえば，D(またはd)と指定するとエンドコードとしてコントロールD(&H04)が使用される。またこのパラメータを省略すると，エンドコードの送受信処理は行なわない。

〔注〕
● 通信パラメータのうちボーレート，パリティ，データ長，ストップビット長は省略できないが，それ以降のパラメータは省略できる。ただし，途中のパラメータを省略して次のパラメータを指定することはできない。
● 漢字の送受信はデータ長8ビットのときに可能（シフトJISコードを使うことになる)。また，カタカナの送受信はSI/SOコードを使うことによってデータ長が7ビットのときでも可能となる（カナの表現方法指定がSであれば自動的にカナに変換する)。

| 各USR関数について | | | |
|---|---|---|---|
| USR0：OPEN | オープンする（パラメータ先頭に O,I,C のいずれかを付加すること） | USR5：PRINT | 文字列＋CR(＋LF）を出力する |
| USR1：CLOSE | クローズする | USR6：LOC | バッファ内のデータ数を返す |
| USR2：LINPUT | 1行入力する（CRもしくはCR＋LFまで） | USR7：EOF | エンドコードを入力したかどうかを返す |
| USR3：INPUTN | N個の文字を入力する（個数は引数の文字列の先頭の ASCII コード） | USR8：PGETC1 | 生データを1文字入力する（カナの処理などはしない） |
| USR4：PRINTS | 文字列を出力する | USR9：PPUTC1 | 生データを1文字出力する（カナの処理などはしない） |

注）USR2, USR3は，返す文字列の先頭のASCIIコードで入力されたデータ数を示す。

そして**リスト 7-1** ＝ソースリスト，**リスト 7-2** ＝ダンプリスト（打ち込み確認には付録Aを参照のこと），**リスト 7-3** ＝サンプルプログラム，**リスト 7-4** ＝簡単なターミナルソフトである（ただし使いものにはならないので，あくまでサンプルにすぎない）。リスト 7-1 の最初の方を見ると分かるだろうが，Z80 SIO，CTC のアドレスがそれぞれ $1F98_H$ と $1FA8_H$ となっている。これは CZ-8BM2 のアドレス設定スイッチが，工場出荷時の 2 - 3 側になっている場合のものである。スイッチが 1 - 2 側もしくは turbo の場合には，それぞれ $1F90_H$，$1FA0_H$ とすればよい。後は説明しても空しいのでやめておく。

**リスト 7-1　通信プログラムのソースリスト**

```
                                    .Z80
                                    .PHASE   0EA00H
                            ;
0330                        QBREAK   EQU      0330H
1FCA                        BBREAK   EQU      1FCAH
                            ;
1F98                        ZSIO     EQU      1F98H    ;IF NOT,1F98H
1FA8                        ZCTC     EQU      1FA8H    ;IF NOT,1FA8H
                            ;
005C                        INTRSV   EQU      5CH      ;INT VECTOR
0040                        QMAX     EQU      64       ;QUE SIZE
0080                        QMAXX    EQU      QMAX+64  ;Q SIZE+ALPHA
                            ;
000D                        CR       EQU      0DH
000A                        LF       EQU      0AH
000E                        SO       EQU      0EH
000F                        SI       EQU      0FH
0011                        XON      EQU      11H
0013                        XOFF     EQU      13H      ;CTRL CODES
                            ;
                            ;BAU RATE         0-6
                            ;PARITY           E,O,N
                            ;DATA BIT LEN     5,6,7,8
                            ;STOP BIT LEN     1,2,3
                            ;CONTROLE         X,R,N,""
                            ;KANA MODE        S,N,""
                            ;CR,LF  SEND      C,L,""
                            ;CR,LF  RECEIVE   C,L,""
                            ;KANJI            J,N,""   ;DUMMY
                            ;END CODE         '@'-'_'  ;PARAMS
                            ;
                            ;JUMP TABLE
EA00     C3 EA1E                     JP       OPEN     ;+0
                            ;I,O or C + PARAMETER
EA03     C3 EB84                     JP       CLOSE    ;+3
                            ;TX ENDCODE
                            ;
EA06     C3 ECDD                     JP       LINPUT   ;+6
                            ;UNTIL CR (+LF)
EA09     C3 ED35                     JP       INPUTN   ;+9
                            ;INPUT n CHAR
```

```
EA0C    C3 ED68          ;
                                 JP      PRINTS  ;+12
                         ;TX STRING
EA0F    C3 ED56                  JP      PRINT   ;+15
                         ;TX STRING + CR (+LF)
                         ;
EA12    C3 EB98                  JP      LOC     ;+18
                         ;RETURN BUFF SIZE
EA15    C3 EBB9                  JP      EOF     ;+21
                         ;EOF OR NOT
                         ;
EA18    C3 EC9A                  JP      PGETC1  ;+24
                         ;RIMITIVE GET CHAR
EA1B    C3 ED90                  JP      PPUTC1  ;+27
                         ;PRIMITIVE PUT CHAR
                         ;
EA1E    3A EE60          OPEN:   LD      A,(OPENF)
EA21    B7                       OR      A
EA22    28 07                    JR      Z,OPENGO
                         ;
                         ;REOPEN THEN CLOSE
EA24    D5                       PUSH    DE
EA25    C5                       PUSH    BC
EA26    CD EB84                  CALL    CLOSE
EA29    C1                       POP     BC
EA2A    D1                       POP     DE
                         ;
EA2B    3E FF            OPENGO: LD      A,0FFH
EA2D    32 EE60                  LD      (OPENF),A       ;SET FLAG
                         ;
EA30    3D               OPENW:  DEC     A
EA31    20 FD                    JR      NZ,OPENW        ;WAIT
                         ;
EA33    F3                       DI
EA34    21 EBC6                  LD      HL,INTRSR
EA37    22 005C                  LD      (INTRSV),HL
EA3A    FB                       EI              ;SET VECTOR
                         ;
EA3B    EB                       EX      DE,HL
EA3C    78                       LD      A,B     ;COPY LEN
EA3D    B7                       OR      A
EA3E    CA EE26                  JP      Z,ERR   ;NULL STR
EA41    05                       DEC     B
EA42    7E                       LD      A,(HL)  ;1 CHAR
EA43    23                       INC     HL
EA44    FE 49                    CP      'I'
EA46    28 0B                    JR      Z,OPENOK
EA48    FE 4F                    CP      'O'
EA4A    28 07                    JR      Z,OPENOK
EA4C    FE 43                    CP      'C'
EA4E    28 03                    JR      Z,OPENOK
EA50    C3 EE26                  JP      ERR
                         ;
EA53    32 EE4B          OPENOK: LD      (IOC),A ;I,O or C
EA56    78                       LD      A,B     ;GET COUNT
EA57    D6 04                    SUB     4
EA59    DA EE26                  JP      C,ERR
EA5C    FE 07                    CP      7
EA5E    D2 EE26                  JP      NC,ERR
EA61    4F                       LD      C,A     ;COUNTER
EA62    7E                       LD      A,(HL)
EA63    23                       INC     HL
                         ;
                         ;BAU RATE        0-6
EA64    D6 30                    SUB     '0'
EA66    FE 07                    CP      7
EA68    D2 EE26                  JP      NC,ERR
EA6B    16 D0                    LD      D,208
EA6D    D6 02                    SUB     2
EA6F    F5                       PUSH    AF
EA70    30 01                    JR      NC,OPEN0
EA72    AF                       XOR     A
EA73    B7               OPEN0:  OR      A
EA74    28 05                    JR      Z,OPEN1
EA76    CB 3A                    SRL     D       ;D=D/2
EA78    3D                       DEC     A
EA79    18 F8                    JR      OPEN0
                         ;
EA7B    7A               OPEN1:  LD      A,D
EA7C    32 EE2E                  LD      (CTR),A
```

```
                        ;
EA7F    F1                      POP     AF
EA80    16 40                   LD      D,40H           ;*16
EA82    30 07                   JR      NC,OPEN2
EA84    16 80                   LD      D,80H           ;*32
EA86    3C                      INC     A
EA87    28 02                   JR      Z,OPEN2
EA89    16 C0                   LD      D,0C0H          ;*64
                        ;
EA8B    5A              OPEN2:  LD      E,D             ;KEEP
                        ;
                        ;PARITY          E,O,N
EA8C    7E                      LD      A,(HL)
EA8D    23                      INC     HL
EA8E    16 00                   LD      D,0
EA90    FE 4E                   CP      'N'
EA92    28 0B                   JR      Z,OPEN3
EA94    14                      INC     D               ;D=1
EA95    FE 4F                   CP      'O'
EA97    28 06                   JR      Z,OPEN3
EA99    14                      INC     D               ;D=2
EA9A    FE 45                   CP      'E'
EA9C    C2 EE26                 JP      NZ,ERR
EA9F    7B              OPEN3:  LD      A,E
EAA0    B2                      OR      D
EAA1    5F                      LD      E,A             ;KEEP
                        ;
                        ;DETA BIT LEN    5,6,7,8
EAA2    7E                      LD      A,(HL)
EAA3    23                      INC     HL
EAA4    D6 35                   SUB     '5'
EAA6    DA EE26                 JP      C,ERR
EAA9    FE 04                   CP      4
EAAB    D2 EE26                 JP      NC,ERR
                        ;A=0-3
EAAE    F5                      PUSH    AF
EAAF    16 E0                   LD      D,11100000B
EAB1    B7                      OR      A
EAB2    28 05                   JR      Z,OPEN35
EAB4    CB 22           MLOOP:  SLA     D
EAB6    3D                      DEC     A
EAB7    20 FB                   JR      NZ,MLOOP
                        ;
EAB9    7A              OPEN35: LD      A,D
EABA    2F                      CPL
EABB    32 EE61                 LD      (MASK),A
EABE    F1                      POP     AF
EABF    B7                      OR      A
EAC0    EA EAC5                 JP      PE,OPEN4
EAC3    EE 03                   XOR     11B
EAC5    0F              OPEN4:  RRCA
EAC6    0F                      RRCA
EAC7    F6 01                   OR      1
EAC9    32 EE5A                 LD      (WR3+1),A       ;RCV
EACC    0F                      RRCA
EACD    F6 80                   OR      80H
EACF    F6 02                   OR      02H
EAD1    F6 08                   OR      08H
EAD3    32 EE54                 LD      (WR5+1),A       ;SND
                        ;
                        ;STOP BIT LEN    1,2,3
EAD6    7E                      LD      A,(HL)
EAD7    23                      INC     HL
EAD8    D6 31                   SUB     '1'
EADA    DA EE26                 JP      C,ERR
EADD    FE 03                   CP      3
EADF    D2 EE26                 JP      NC,ERR
EAE2    3C                      INC     A
EAE3    87                      ADD     A,A
EAE4    87                      ADD     A,A     ;A=A<<2
EAE5    B3                      OR      E
EAE6    32 EE52                 LD      (WR4+1),A
                        ;
                        ;CONTROLE        X,R,N,""
                        ;KANA MODE       S,N,""
                        ;CR,LF  SEND     C,L,""
                        ;CR,LF  RECEIVE  C,L,""
                        ;KANJI           J,N,""  ;DUMMY
                        ;END CODE        '@'-'_'
                        ;
EAE9    C5                      PUSH    BC
EAEA    E5                      PUSH    HL
```

```
EAEB    21 EE3F                  LD    HL,PDFLT
EAEE    11 EE45                  LD    DE,PARAM
EAF1    01 0006                  LD    BC,6
EAF4    ED B0                    LDIR           ;COPY DEFAULT
EAF6    EB                       EX    DE,HL    ;DE'=PARAM
EAF7    D9                       EXX
EAF8    E1                       POP   HL
EAF9    C1                       POP   BC
EAFA    11 EE2F                  LD    DE,PLIST
EAFD    AF                       XOR   A        ;FLAG
EAFE    08                       EX    AF,AF'   ;A'=0
EAFF    79                       LD    A,C
EB00    B7                       OR    A
EB01    28 1F                    JR    Z,FINIS
EB03    FE 06                    CP    6
EB05    20 02                    JR    NZ,OPEN6
EB07    0D                       DEC   C        ;C=5
EB08    08                       EX    AF,AF'   ;A'<>0
                        ;
EB09    46              OPEN6:   LD    B,(HL)
EB0A    23                       INC   HL
EB0B    1A              OPEN7:   LD    A,(DE)
EB0C    13                       INC   DE
EB0D    B7                       OR    A        ;SEPARATOR
EB0E    CA EE26                  JP    Z,ERR
EB11    B8                       CP    B
EB12    20 F7                    JR    NZ,OPEN7
                        ;MATCH
EB14    D9                       EXX
EB15    12                       LD    (DE),A
EB16    D9                       EXX
EB17    1A              OPEN75:  LD    A,(DE)
EB18    13                       INC   DE
EB19    B7                       OR    A
EB1A    20 FB                    JR    NZ,OPEN75
EB1C    D9              OPEN8:   EXX
EB1D    13                       INC   DE
EB1E    D9                       EXX
EB1F    0D                       DEC   C
EB20    20 E7                    JR    NZ,OPEN6
                        ;
EB22    08              FINIS:   EX    AF,AF'
EB23    B7                       OR    A
EB24    28 10                    JR    Z,SETZ
EB26    7E                       LD    A,(HL)   ;ENDCODE
EB27    FE 40                    CP    40H
EB29    DA EE26                  JP    C,ERR
EB2C    FE 80                    CP    80H
EB2E    D2 EE26                  JP    NC,ERR
EB31    E6 1F                    AND   1FH
EB33    D9                       EXX
EB34    12                       LD    (DE),A   ;STORE
EB35    D9                       EXX
                        ;
EB36    F3              SETZ:    DI             ;SET ZCTC,ZSIO
EB37    CD EB6A                  CALL  INITBF
EB3A    01 1FA9                  LD    BC,ZCTC+1        ;CH0
EB3D    3E 47                    LD    A,01000111B      ;MODE
EB3F    ED 79                    OUT   (C),A
EB41    3A EE2E                  LD    A,(CTR)
EB44    ED 79                    OUT   (C),A            ;T CONSTANT
                        ;
EB46    21 EE4C                  LD    HL,SIOSA
EB49    01 1F99                  LD    BC,ZSIO+1        ;CH A
EB4C    3E 0F                    LD    A,15
EB4E    CD EB63                  CALL  SETSIO
                        ;
EB51    21 EE5B                  LD    HL,SIOSB
EB54    01 1F9B                  LD    BC,ZSIO+3        ;CH B
EB57    3E 05                    LD    A,5
EB59    CD EB63                  CALL  SETSIO
EB5C    3E FF                    LD    A,0FFH
EB5E    32 EE60                  LD    (OPENF),A
EB61    FB                       EI
EB62    C9                       RET
                        ;
EB63    04              SETSIO:  INC   B
EB64    ED A3                    OUTI
EB66    3D                       DEC   A
EB67    20 FA                    JR    NZ,SETSIO
EB69    C9                       RET
```

```
                                ;
EB6A    AF                      INITBF: XOR     A
EB6B    32 EE6A                         LD      (QLEN),A
EB6E    21 EE6B                         LD      HL,Q0
EB71    22 EE66                         LD      (HEAD),HL
EB74    22 EE68                         LD      (TAIL),HL
                                ;
EB77    32 EE62                         LD      (LASTX),A
EB7A    32 EE63                         LD      (XFLAG),A
EB7D    32 EE64                         LD      (TKIN),A
EB80    32 EE65                         LD      (RKIN),A        ;RESET MODE
EB83    C9                              RET
                                ;
EB84    AF                      CLOSE:  XOR     A
EB85    32 EE60                         LD      (OPENF),A
                                ;
EB88    3A EE4B                         LD      A,(IOC)
EB8B    FE 4F                           CP      'O'
EB8D    C0                              RET     NZ
EB8E    3A EE4A                         LD      A,(ECODE)
EB91    FE 20                           CP      20H
EB93    C8                              RET     Z
EB94    CD EDCD                         CALL    PPUTC   ;TX ECODE
EB97    C9                              RET
                                ;
EB98    3A EE6A                 LOC:    LD      A,(QLEN)
EB9B    77                              LD      (HL),A
EB9C    23                              INC     HL
EB9D    36 00                           LD      (HL),0  ;RET SIZE
EB9F    C9                              RET
                                ;
                                ;BREAK B,A,F
EBA0    3A EE6A                 QEOF:   LD      A,(QLEN)
EBA3    B7                              OR      A
EBA4    C8                              RET     Z       ;EMPTY
EBA5    3A EE4A                         LD      A,(ECODE)
EBA8    FE 20                           CP      20H
EBAA    C8                              RET     Z       ;NO ECODE
EBAB    E5                              PUSH    HL
EBAC    2A EE68                         LD      HL,(TAIL)
EBAF    46                              LD      B,(HL)  ;NOZOKU
EBB0    E1                              POP     HL
EBB1    B8                              CP      B
EBB2    28 02                           JR      Z,QEOF1
EBB4    AF                              XOR     A       ;SET Z
EBB5    C9                              RET
EBB6    F6 FF                   QEOF1:  OR      0FFH    ;SET NZ
EBB8    C9                              RET
                                ;
EBB9    CD EBA0                 EOF:    CALL    QEOF
EBBC    11 0000                         LD      DE,0
EBBF    28 01                           JR      Z,EOF1
EBC1    1B                              DEC     DE
EBC2    73                      EOF1:   LD      (HL),E
EBC3    23                              INC     HL
EBC4    72                              LD      (HL),D
EBC5    C9                              RET
                                ;
EBC6    F3                      INTRSR: DI                      ;INT ROUTINE
EBC7    F5                              PUSH    AF
EBC8    C5                              PUSH    BC
EBC9    D5                              PUSH    DE
EBCA    E5                              PUSH    HL
                                ;
EBCB    01 1F99                         LD      BC,ZSIO+1
                                ;
EBCE    3E 01                   INTR1:  LD      A,1
EBD0    ED 79                           OUT     (C),A   ;RR1
EBD2    ED 78                           IN      A,(C)
                                ;
EBD4    0B                              DEC     BC
EBD5    ED 78                           IN      A,(C)
EBD7    2A EE66                         LD      HL,(HEAD)
EBDA    77                              LD      (HL),A
EBDB    23                              INC     HL
EBDC    E5                              PUSH    HL
EBDD    01 EEEB                         LD      BC,Q1
EBE0    B7                              OR      A
EBE1    ED 42                           SBC     HL,BC
EBE3    E1                              POP     HL
EBE4    20 03                           JR      NZ,INTR2
```

```
EBE6      21 EE6B               LD      HL,Q0           ;QUE !
EBE9      22 EE66       INTR2:  LD      (HEAD),HL
EBEC      21 EE6A               LD      HL,QLEN
EBEF      34                    INC     (HL)
                        ;
EBF0      FE 13                 CP      XOFF
EBF2      20 05                 JR      NZ,INTR3
EBF4      32 EE62               LD      (LASTX),A       ;GET XOFF
EBF7      18 08                 JR      INTR4
EBF9      FE 11         INTR3:  CP      XON
EBFB      20 04                 JR      NZ,INTR4
EBFD      AF                    XOR     A
EBFE      32 EE62               LD      (LASTX),A       ;GET XON
EC01      3A EE6A       INTR4:  LD      A,(QLEN)
EC04      3C                    INC     A
EC05      FE 40                 CP      QMAX
EC07      D4 EC11               CALL    NC,SWAIT        ;SEND WAIT
                        ;
EC0A      E1                    POP     HL
EC0B      D1                    POP     DE
EC0C      C1                    POP     BC
EC0D      F1                    POP     AF
EC0E      FB                    EI
EC0F      ED 4D                 RETI
                        ;
EC11      01 1F99       SWAIT:  LD      BC,ZSIO+1
EC14      3A EE45               LD      A,(PARAM+0)
EC17      FE 4E                 CP      'N'
EC19      C8                    RET     Z       ;NOTHING
EC1A      FE 52                 CP      'R'
EC1C      28 11                 JR      Z,SWAIT1
EC1E      FE 58                 CP      'X'
EC20      C0                    RET     NZ
EC21      3A EE63               LD      A,(XFLAG)
EC24      B7                    OR      A
EC25      C0                    RET     NZ
EC26      3E 13                 LD      A,XOFF  ;X CTRL
EC28      32 EE63               LD      (XFLAG),A
EC2B      CD ED77               CALL    PPUTC0
EC2E      C9                    RET
EC2F      01 1F99       SWAIT1: LD      BC,ZSIO+1
EC32      3E 05                 LD      A,5
EC34      ED 79                 OUT     (C),A
EC36      3A EE54               LD      A,(WR5+1)
EC39      E6 FD                 AND     11111101B       ;RES RTS
EC3B      ED 79                 OUT     (C),A
EC3D      C9                    RET
                        ;
EC3E      3A EE45       SOK:    LD      A,(PARAM+0)
EC41      FE 4E                 CP      'N'
EC43      C8                    RET     Z
                        ;
EC44      FE 52                 CP      'R'
EC46      28 12                 JR      Z,SOK1
EC48      FE 58                 CP      'X'
EC4A      C0                    RET     NZ
EC4B      3A EE63               LD      A,(XFLAG)
EC4E      B7                    OR      A
EC4F      C8                    RET     Z
EC50      AF                    XOR     A
EC51      32 EE63               LD      (XFLAG),A
EC54      3E 11                 LD      A,XON
EC56      CD ED77               CALL    PPUTC0
EC59      C9                    RET
EC5A      01 1F99       SOK1:   LD      BC,ZSIO+1
EC5D      3E 05                 LD      A,5
EC5F      ED 79                 OUT     (C),A
EC61      3A EE54               LD      A,(WR5+1)
EC64      F6 02                 OR      00000010B       ;SET RTS
EC66      ED 79                 OUT     (C),A
EC68      C9                    RET
                        ;
EC69      CD ECA4       XGETC:  CALL    PGETC   ;GET CHAR
EC6C      47                    LD      B,A     ;SAVE
EC6D      3A EE46               LD      A,(PARAM+1)
EC70      FE 53                 CP      'S'
EC72      20 24                 JR      NZ,XGETC4
EC74      3A EE61               LD      A,(MASK)
EC77      FE 7F                 CP      1111111B
EC79      20 1D                 JR      NZ,XGETC4
                        ;
```

```
                                   ;S AND 7 BITS
EC7B    3A EE65                         LD      A,(RKIN)
EC7E    B7                              OR      A
EC7F    78                              LD      A,B
EC80    20 06                           JR      NZ,XGETC1
EC82    FE 0E                           CP      SO      ;TO KANA MODE
EC84    28 07                           JR      Z,XGETC2
EC86    18 10                           JR      XGETC4
                                   ;
EC88    FE 0F                      XGETC1: CP   SI
EC8A    20 06                           JR      NZ,XGETC3
EC8C    AF                              XOR     A
EC8D    32 EE65                    XGETC2: LD   (RKIN),A        ;CLEAR
EC90    18 D7                           JR      XGETC           ;AGAIN
                                   ;
EC92    FE 20                      XGETC3: CP   0A0H-080H
EC94    38 02                           JR      C,XGETC4
EC96    CB F8                           SET     7,B     ;TO KANA
EC98    78                         XGETC4: LD   A,B
EC99    C9                              RET
                                   ;
EC9A    E5                         PGETC1: PUSH HL      ;RETURN CHAR
EC9B    CD ECA4                         CALL    PGETC
EC9E    E1                              POP     HL
EC9F    77                              LD      (HL),A
ECA0    23                              INC     HL
ECA1    36 00                           LD      (HL),0
ECA3    C9                              RET
                                   ;
ECA4    CD 0330                    PGETC:  CALL QBREAK
ECA7    CA EE2A                         JP      Z,BREAK ;BREAK?
ECAA    3A EE6A                         LD      A,(QLEN)
ECAD    B7                              OR      A
ECAE    28 F4                           JR      Z,PGETC
                                   ;
ECB0    F3                              DI
ECB1    3A EE6A                         LD      A,(QLEN)        ;RE-GET
ECB4    5F                              LD      E,A             ;SAVE QLEN
ECB5    2A EE68                         LD      HL,(TAIL)
ECB8    3A EE61                         LD      A,(MASK)
ECBB    A6                              AND     (HL)
ECBC    57                              LD      D,A     ;GET DATA
ECBD    23                              INC     HL
ECBE    E5                              PUSH    HL
ECBF    01 EEEB                         LD      BC,Q1
ECC2    B7                              OR      A
ECC3    ED 42                           SBC     HL,BC
ECC5    E1                              POP     HL
ECC6    20 03                           JR      NZ,GETC1
ECC8    21 EE6B                         LD      HL,Q0           ;QUE !
ECCB    22 EE68                    GETC1:  LD   (TAIL),HL
ECCE    7B                              LD      A,E             ;GET QLEN
ECCF    3D                              DEC     A
ECD0    32 EE6A                         LD      (QLEN),A
ECD3    FE 30                           CP      QMAX-16
ECD5    D5                              PUSH    DE
ECD6    DC EC3E                         CALL    C,SOK           ;SEND OK
ECD9    FB                              EI
ECDA    D1                              POP     DE
ECDB    7A                              LD      A,D
ECDC    C9                              RET
                                   ;
ECDD    EB                         LINPUT: EX   DE,HL
ECDE    E5                              PUSH    HL
ECDF    23                              INC     HL
ECE0    78                              LD      A,B
ECE1    FE 02                           CP      2
ECE3    DA EE26                         JP      C,ERR   ;NO SPACE
ECE6    05                              DEC     B       ;DEC COUNTER
ECE7    0E 00                           LD      C,0
                                   ;
ECE9    E5                         LIP1:   PUSH HL
ECEA    C5                              PUSH    BC      ;SAVE COUNTER
ECEB    CD 0330                    LIP1L:  CALL QBREAK  ;BREAK?
ECEE    CA EE2A                         JP      Z,BREAK
ECF1    3A EE6A                         LD      A,(QLEN)
ECF4    B7                              OR      A
ECF5    28 F4                           JR      Z,LIP1L
ECF7    CD EBA0                         CALL    QEOF
ECFA    28 0A                           JR      Z,LIP1L1
                                   ;GET EOF
```

```
ECFC    C1                  POP     BC
ECFD    E1                  POP     HL
ECFE    79                  LD      A,C
ECFF    FE 0D               CP      CR
ED01    20 2A               JR      NZ,LIPL2
ED03    2B                  DEC     HL
ED04    18 27               JR      LIPL2   ;SUTE CR
                    ;
ED06    CD EC69     LIP1L1: CALL    XGETC
ED09    C1                  POP     BC
ED0A    E1                  POP     HL
                    ;1 LINE?
ED0B    F5                  PUSH    AF
ED0C    3A EE48             LD      A,(PARAM+3)
ED0F    FE 4C               CP      'L'
ED11    28 07               JR      Z,LIPL1
ED13    F1                  POP     AF
ED14    FE 0D               CP      CR
ED16    28 15               JR      Z,LIPL2
ED18    18 0E               JR      LIP2
ED1A    79          LIPL1:  LD      A,C
ED1B    FE 0D               CP      CR
ED1D    20 08               JR      NZ,LIPL3
ED1F    F1                  POP     AF
ED20    FE 0A               CP      LF
ED22    20 04               JR      NZ,LIP2
ED24    2B                  DEC     HL
ED25    18 06               JR      LIPL2
                    ;
ED27    F1          LIPL3:  POP     AF
ED28    77          LIP2:   LD      (HL),A
ED29    4E          LIP3:   LD      C,(HL)  ;LAST CHAR
ED2A    23                  INC     HL
ED2B    10 BC               DJNZ    LIP1    ;FULL?
ED2D    D1          LIPL2:  POP     DE
ED2E    B7                  OR      A
ED2F    ED 52               SBC     HL,DE
ED31    EB                  EX      DE,HL
ED32    1D                  DEC     E
ED33    73                  LD      (HL),E  ;STORE LEN
ED34    C9                  RET
                    ;
                    ;INPUT n CHAR
ED35    D5          INPUTN: PUSH    DE      ;SAVE ADDR
ED36    EB                  EX      DE,HL
ED37    7E                  LD      A,(HL)
ED38    23                  INC     HL
ED39    B8                  CP      B
ED3A    D2 EE26             JP      NC,ERR  ;TOO SHORT
                    ;A<B
ED3D    05                  DEC     B
ED3E    B7                  OR      A
ED3F    28 12               JR      Z,IPN2  ;REQUEST 0 CHAR
ED41    47                  LD      B,A     ;COUNTER
ED42    0E 00               LD      C,0     ;LEN
                    ;
ED44    C5          IPN1:   PUSH    BC
ED45    E5                  PUSH    HL
ED46    CD EC69             CALL    XGETC
ED49    E1                  POP     HL
ED4A    C1                  POP     BC
ED4B    77                  LD      (HL),A  ;STORE CODE
ED4C    23                  INC     HL
ED4D    0C                  INC     C
ED4E    10 F4               DJNZ    IPN1
ED50    E1                  POP     HL      ;TOP ADDR.
ED51    71                  LD      (HL),C  ;STORE LEN
ED52    C9                  RET
                    ;
ED53    D1          IPN2:   POP     DE
ED54    12                  LD      (DE),A  ;STORE 0
ED55    C9                  RET
                    ;
ED56    CD ED68     PRINT:  CALL    PRINTS
ED59    3E 0D               LD      A,CR
ED5B    CD EDCD             CALL    PPUTC
ED5E    3A EE47             LD      A,(PARAM+2)
ED61    FE 4C               CP      'L'
ED63    C0                  RET     NZ      ;'C'
ED64    3E 0A               LD      A,LF    ;'L'
ED66    18 65               JR      PPUTC
```

```
                                        ;
ED68    78                              PRINTS: LD      A,B
ED69    B7                                      OR      A       ;CK LEN
ED6A    C8                                      RET     Z
ED6B    1A                              PUTSL:  LD      A,(DE)
ED6C    13                                      INC     DE
ED6D    C5                                      PUSH    BC
ED6E    D5                                      PUSH    DE
ED6F    CD ED93                                 CALL    PUTC
ED72    D1                                      POP     DE
ED73    C1                                      POP     BC
ED74    10 F5                                   DJNZ    PUTSL
ED76    C9                                      RET
                                        ;
ED77    F5                              PPUTC0: PUSH    AF
ED78    01 1F99                                 LD      BC,ZSIO+1
ED7B    3E 10                           PUTCL0: LD      A,10H   ;RESET STAT
ED7D    ED 79                                   OUT     (C),A
ED7F    ED 78                                   IN      A,(C)   ;GET RR0
ED81    CB 57                                   BIT     2,A     ;TX BUFF
ED83    28 F6                                   JR      Z,PUTCL0
                                        ;
ED85    CD EE09                                 CALL    CHTXOK  ;TX OK?
ED88    28 62                                   JR      Z,PUTOK
ED8A    3E 10                                   LD      A,10H   ;STAT RESET
ED8C    ED 79                                   OUT     (C),A
ED8E    18 EB                                   JR      PUTCL0  ;LOOP
                                        ;
ED90    7E                              PPUTC1: LD      A,(HL)
ED91    18 3A                                   JR      PPUTC
                                        ;
ED93    47                              PUTC:   LD      B,A
ED94    3A EE46                                 LD      A,(PARAM+1)
ED97    FE 53                                   CP      'S'
ED99    20 31                                   JR      NZ,PUTC0
                                        ;
ED9B    3A EE61                                 LD      A,(MASK)
ED9E    FE 7F                                   CP      1111111B
EDA0    20 2A                                   JR      NZ,PUTC0
                                        ;
                                        ;'S' AND 7 BITS
EDA2    CB 78                                   BIT     7,B
EDA4    28 15                                   JR      Z,NKANA ;NOT KANA
EDA6    3A EE64                                 LD      A,(TKIN)
EDA9    B7                                      OR      A
EDAA    20 0A                                   JR      NZ,IKANA
EDAC    C5                                      PUSH    BC      ;SAVE IT
EDAD    3E 0F                                   LD      A,SI    ;SHIFT IN
EDAF    CD EDCD                                 CALL    PPUTC
EDB2    32 EE64                                 LD      (TKIN),A
EDB5    C1                                      POP     BC
                                        ;
EDB6    78                              IKANA:  LD      A,B
EDB7    E6 7F                                   AND     7FH     ;SEND 7 BITS
EDB9    18 12                                   JR      PPUTC
                                        ;
EDBB    3A EE64                         NKANA:  LD      A,(TKIN)
EDBE    B7                                      OR      A
EDBF    28 0B                                   JR      Z,PUTC0 ;NO KANA MODE
                                        ;
EDC1    C5                                      PUSH    BC      ;SAVE IT
EDC2    3E 0E                                   LD      A,SO    ;SHIFT OUT
EDC4    CD EDCD                                 CALL    PPUTC   ;PUT ANY HOW
EDC7    AF                                      XOR     A
EDC8    32 EE64                                 LD      (TKIN),A
EDCB    C1                                      POP     BC      ;B=IT
                                        ;
EDCC    78                              PUTC0:  LD      A,B     ;DATA TO TX
EDCD    F5                              PPUTC:  PUSH    AF
EDCE    01 1F99                                 LD      BC,ZSIO+1
EDD1    CD 0330                         PUTCL:  CALL    QBREAK  ;BREAK?
EDD4    CA EE2A                                 JP      Z,BREAK
EDD7    3E 10                                   LD      A,10H   ;RESET STAT
EDD9    ED 79                                   OUT     (C),A
EDDB    ED 78                                   IN      A,(C)   ;GET RR0
EDDD    CB 57                                   BIT     2,A     ;TX BUFF
EDDF    28 F0                                   JR      Z,PUTCL
                                        ;
EDE1    CD EE09                                 CALL    CHTXOK  ;TX OK?
EDE4    28 06                                   JR      Z,PUTOK
EDE6    3E 10                                   LD      A,10H   ;STAT RESET
```

```
EDE8    ED 79                   OUT     (C),A
EDEA    18 E5                   JR      PUTCL     ;LOOP
                        ;
EDEC    0B              PUTOK:  DEC     BC        ;DATA PORT
EDED    D1                      POP     DE        ;D=CHAR
EDEE    3A EE61                 LD      A,(MASK)
EDF1    A2                      AND     D
EDF2    ED 79                   OUT     (C),A     ;TX DATA
                        ;
EDF4    3A EE6A                 LD      A,(QLEN)
EDF7    B7                      OR      A         ;EMPTY?
EDF8    C8                      RET     Z
                        ;
                        ;DROP XON,XOFF
EDF9    2A EE68         PEND0:  LD      HL,(TAIL)
EDFC    7E                      LD      A,(HL)    ;NOZOKU
EDFD    FE 11                   CP      XON
EDFF    28 03                   JR      Z,PEND1
EE01    FE 13                   CP      XOFF
EE03    C0                      RET     NZ
EE04    CD EC9A         PEND1:  CALL    PGETC1    ;DROP
EE07    18 F0                   JR      PEND0
                        ;
EE09    57              CHTXOK: LD      D,A       ;SAVE RR0
EE0A    3A EE45                 LD      A,(PARAM+0)
EE0D    FE 4E                   CP      'N'       ;NO CON
EE0F    C8                      RET     Z
EE10    FE 52                   CP      'R'       ;RTS
EE12    20 05                   JR      NZ,CHX
EE14    7A                      LD      A,D
EE15    2F                      CPL               ;REVERSE
EE16    CB 6F                   BIT     5,A       ;RTS
EE18    C9                      RET
                        ;
EE19    FE 58           CHX:    CP      'X'
EE1B    28 02                   JR      Z,CHX1
EE1D    AF                      XOR     A
EE1E    C9                      RET
EE1F    F3              CHX1:   DI                ;ENGIMON
EE20    3A EE62                 LD      A,(LASTX)
EE23    B7                      OR      A
EE24    FB                      EI
EE25    C9                      RET
                        ;
EE26    3E 05           ERR:    LD      A,5       ;ILL FUNC
EE28    DD E9                   JP      (IX)
                        ;
EE2A    F3              BREAK:  DI
EE2B    C3 1FCA                 JP      BBREAK
                        ;
EE2E                    CTR:    DS      1         ;CTC COUNTER
                        ;
EE2F    58 52 4E 00     PLIST:  DB      'XRN',0
EE33    53 4E 00                DB      'SN',0
EE36    43 4C 00                DB      'CL',0
EE39    43 4C 00                DB      'CL',0
EE3C    4A 4E 00                DB      'JN',0
                        ;
EE3F    4E 4E 4C 4C     PDFLT:  DB      'NNLLN',20H
EE43    4E 20
                        ;
EE45                    PARAM:  DS      5         ;P AREA
EE4A                    ECODE:  DS      1         ;5+1=6
                        ;
EE4B                    IOC:    DS      1         ;OPEN MODE FLAG
                        ;
EE4C    18              SIOSA:  DB      00011000B         ;WR0
EE4D    01 10           WR1:    DB      1,10H
EE4F    02 00           WR2:    DB      2,00H
EE51    04 00           WR4:    DB      4,00H
EE53    05 00           WR5:    DB      5,00H
EE55    06 00           WR6:    DB      6,00H
EE57    07 00           WR7:    DB      7,00H
EE59    03 00           WR3:    DB      3,00H
                        ;
EE5B    18              SIOSB:  DB      18H
EE5C    01 00                   DB      1,00H
EE5E    02 5C                   DB      2,INTRSV
                        ;
EE60    00              OPENF:  DB      0         ;OPEB FLAG
                        ;
```

```
EE61                          MASK:   DS      1         ;DATA MASK
EE62                          LASTX:  DS      1         ;R FLAG
EE63                          XFLAG:  DS      1         ;T FLAG
                              ;
EE64                          TKIN:   DS      1         ;R KANA FLAG
EE65                          RKIN:   DS      1         ;T KANA FLAG
                              ;
EE66                          HEAD:   DS      2         ;QUE HEAD
EE68                          TAIL:   DS      2         ;QUE TAIL
EE6A                          QLEN:   DS      1         ;Q LENGTH
EE6B                          Q0:     DS      QMAXX
EEEB                          Q1:
                              ;
                                      END
```

リスト 7-2 通信プログラムのダンプリスト

```
EA00 C3 1E EA C3 84 EB C3 DD : 9D
EA08 EC C3 35 ED C3 68 ED C3 : AC
EA10 56 ED C3 98 EB C3 B9 EB : F0
EA18 C3 9A EC C3 90 ED 3A 60 : 23
EA20 EE B7 28 07 D5 C5 CD 84 : BF
EA28 EB C1 D1 3E FF 32 60 EE : 3A
EA30 3D 20 FD F3 21 C6 EB 22 : 41
EA38 5C 00 FB EB 78 B7 CA 26 : 61
EA40 EE 05 7E 23 FE 49 28 0B : 0E
EA48 FE 4F 28 07 FE 43 28 03 : E8
EA50 C3 26 EE 32 4B EE 78 D6 : 90
EA58 04 DA 26 EE FE 07 D2 26 : EF
EA60 EE 4F 7E 23 D6 30 FE 07 : E9
EA68 D2 26 EE 16 D0 D6 02 F5 : 99
EA70 30 01 AF B7 28 05 CB 3A : C9
EA78 3D 18 F8 7A 32 2E EE F1 : 06
--------------------------------
SUM: 1A E2 8C E2 74 31 D8 D6 6C5A

EA80 16 40 30 07 16 80 3C 28 : 87
EA88 02 16 C0 5A 7E 23 16 00 : E9
EA90 FE 4E 28 0B 14 FE 4F 28 : 08
EA98 06 14 FE 45 C2 26 EE 7B : AE
EAA0 B2 5F 7E 23 D6 35 DA 26 : BD
EAA8 EE FE 04 D2 26 EE F5 16 : E1
EAB0 E0 B7 28 05 CB 22 3D 20 : 0E
EAB8 FB 7A 2F 32 61 EE F1 B7 : CD
EAC0 EA C5 EA EE 03 0F 0F F6 : 9E
EAC8 01 32 5A EE 0F F6 80 F6 : F6
EAD0 02 F6 08 32 54 EE 7E 23 : 15
EAD8 D6 31 DA 26 EE FE 03 D2 : C8
EAE0 26 EE 3C 87 87 B3 32 52 : 95
EAE8 EE C5 E5 21 3F EE 11 45 : 3C
EAF0 EE 01 06 00 ED B0 EB D9 : 56
EAF8 E1 C1 11 2F EE AF 08 79 : 00
--------------------------------
SUM: 3D D9 4D E8 87 EB D2 A8 CC3B

EB00 B7 28 1F FE 06 20 02 0D : 31
EB08 08 46 23 1A 13 B7 CA 26 : 45
EB10 EE B8 20 F7 D9 12 D9 1A : 9B
EB18 13 B7 20 FB D9 13 D9 0D : B7
EB20 20 E7 08 B7 28 10 7E FE : 7A
EB28 40 DA 26 EE FE 80 D2 26 : A4
EB30 EE E6 1F D9 12 D9 F3 CD : 77
EB38 6A EB 01 A9 1F 3E 47 ED : 90
EB40 79 3A 2E EE ED 79 21 4C : A2
EB48 EE 01 99 1F 3E 0F CD 63 : 24
EB50 EB 21 5B EE 01 9B 1F 3E : 4E
EB58 05 CD 63 EB 3E FF 32 60 : EF
EB60 EE FB C9 04 ED A3 3D 20 : A3
EB68 FA C9 AF 32 6A EE 21 6B : 88
```

```
EB70 EE 22 66 EE 22 68 EE 32 : 0E
EB78 62 EE 32 63 EE 32 64 EE : 57
--------------------------------
SUM: 07 6C 65 9E F3 F0 F7 30 1771

EB80 32 65 EE C9 AF 32 60 EE : 7D
EB88 3A 4B EE FE 4F C0 3A 4A : 04
EB90 EE FE 20 C8 CD CD ED C9 : 24
EB98 3A 6A EE 77 23 36 00 C9 : 2B
EBA0 3A 6A EE B7 C8 3A 4A EE : 83
EBA8 FE 20 C8 E5 2A 68 EE 46 : 91
EBB0 E1 B8 28 02 AF C9 F6 FF : 30
EBB8 C9 CD A0 EB 11 00 00 28 : 5A
EBC0 01 1B 73 23 72 C9 F3 F5 : D5
EBC8 C5 D5 E5 01 99 1F 3E 01 : 77
EBD0 ED 79 ED 78 0B ED 78 2A : 65
EBD8 66 EE 77 23 E5 01 EB EE : AD
EBE0 B7 ED 42 E1 20 03 21 6B : 76
EBE8 EE 22 66 EE 21 6A EE 34 : 11
EBF0 FE 13 20 05 32 62 EE 18 : D0
EBF8 08 FE 11 20 04 AF 32 62 : 7E
--------------------------------
SUM: 3A 9E FD 42 12 B4 78 4C 99CD

EC00 EE 3A 6A EE 3C FE 40 D4 : CE
EC08 11 EC E1 D1 C1 F1 FB ED : 49
EC10 4D 01 99 1F 3A 45 EE FE : 71
EC18 4E C8 FE 52 28 11 FE 58 : F5
EC20 C0 3A 63 EE B7 C0 3E 13 : 13
EC28 32 63 EE CD 77 ED C9 01 : 7E
EC30 99 1F 3E 05 ED 79 3A 54 : EF
EC38 EE E6 FD ED 79 C9 3A 45 : 7F
EC40 EE FE 4E C8 FE 52 28 12 : 8C
EC48 FE 58 C0 3A 63 EE B7 C8 : 20
EC50 AF 32 63 EE 3E 11 CD 77 : C5
EC58 ED C9 01 99 1F 3E 05 ED : 9F
EC60 79 3A 54 EE F6 02 ED 79 : 53
EC68 C9 CD A4 EC 47 3A 46 EE : DB
EC70 FE 53 20 24 3A 61 EE FE : 1C
EC78 7F 20 1D 3A 65 EE B7 78 : 78
--------------------------------
SUM: 5A 5C 15 9E 8D 4E 2B DF 9564

EC80 20 06 FE 0E 28 07 18 10 : 89
EC88 FE 0F 20 06 AF 32 65 EE : 67
EC90 18 D7 FE 20 38 02 CB F8 : 0A
EC98 78 C9 E5 CD A4 EC E1 77 : DB
ECA0 23 36 00 C9 CD 30 03 CA : EC
ECA8 2A EE 3A 6A EE B7 28 F4 : 7D
ECB0 F3 3A 6A EE 5F 2A 68 EE : 64
ECB8 3A 61 EE A6 57 23 E5 01 : 8F
ECC0 EB EE B7 ED 42 E1 20 03 : C3
```

```
ECC8 21 6B EE 22 68 EE 7B 3D : AA
ECD0 32 6A EE FE 30 D5 DC 3E : A7
ECD8 EC FB D1 7A C9 EB E5 23 : EE
ECE0 78 FE 02 DA 26 EE 05 0E : 79
ECE8 00 E5 C5 CD 30 03 CA 2A : 9E
ECF0 EE 3A 6A EE B7 28 F4 CD : 20
ECF8 A0 EB 28 0A C1 E1 79 FE : D6
--------------------------------
SUM: 58 3A 50 EE 95 E4 39 BE CDF1

ED00 0D 20 2A 2B 18 27 CD 69 : F7
ED08 EC C1 E1 F5 3A 48 EE FE : F1
ED10 4C 28 07 F1 FE 0D 28 15 : B4
ED18 18 0E 79 FE 0D 20 08 F1 : C3
ED20 FE 0A 20 04 2B 18 06 F1 : 66
ED28 77 4E 23 10 BC D1 B7 ED : 29
ED30 52 EB 1D 73 C9 D5 EB 7E : D4
ED38 23 B8 D2 26 EE 05 B7 28 : A5
ED40 12 47 0E 00 C5 E5 CD 69 : 47
ED48 EC E1 C1 77 23 0C 10 F4 : 38
ED50 E1 71 C9 D1 12 C9 CD 68 : FC
ED58 ED 3E 0D CD CD ED 3A 47 : 40
ED60 EE FE 4C C0 3E 0A 18 65 : BD
ED68 78 B7 C8 1A 13 C5 D5 CD : 8B
ED70 93 ED D1 C1 10 F5 C9 F5 : D5
ED78 01 99 1F 3E 10 ED 79 ED : 5A
--------------------------------
SUM: 0D 24 66 AA 33 B7 5D 11 8D3D

ED80 78 CB 57 28 F6 CD 09 EE : 7C
ED88 28 62 3E 10 ED 79 18 EB : 41
ED90 7E 18 3A 47 3A 46 EE FE : 83
ED98 53 20 31 3A 61 EE FE 7F : AA
EDA0 20 2A CB 78 28 15 3A 64 : 68
EDA8 EE B7 20 0A C5 3E 0F CD : AE
EDB0 CD ED 32 64 EE C1 78 E6 : 5D
EDB8 7F 18 12 3A 64 EE B7 28 : 14
EDC0 0B C5 3E 0E CD CD ED AF : 52
EDC8 32 64 EE C1 78 F5 01 99 : 4C
EDD0 1F CD 30 03 CA 2A EE 3E : 3F
EDD8 10 ED 79 ED 78 CB 57 28 : 25
EDE0 F0 CD 09 EE 28 06 3E 10 : 30
EDE8 ED 79 18 E5 0B D1 3A 61 : DA
EDF0 EE A2 ED 79 3A 6A EE B7 : 3F
EDF8 C8 2A 68 EE 7E FE 11 28 : FD
--------------------------------
SUM: CA 40 7A D2 2F 72 2F 93 6DF0

EE00 03 FE 13 C0 CD 9A EC 18 : 3F
EE08 F0 57 3A 45 EE FE 4E C8 : C8
EE10 FE 52 20 05 7A 2F CB 6F : 58
EE18 C9 FE 58 28 02 AF C9 F3 : B4
EE20 3A 62 EE B7 FB C9 3E 05 : 48
EE28 DD E9 F3 C3 CA 1F 15 58 : D2
EE30 52 4E 00 53 4E 00 43 4C : D0
EE38 00 43 4C 00 4A 4E 00 4E : 75
EE40 4E 4C 4C 4E 20 CD 83 21 : C5
EE48 47 C3 B3 20 18 01 10 02 : 08
EE50 00 04 00 05 00 06 00 07 : 16
EE58 00 03 00 18 01 00 02 5C : 7A
EE60 00                      : 00
--------------------------------
SUM: B8 97 F1 8A CD 80 F9 BF E0FD
```

リスト 7-3 サンプルプログラム

```
100 CLEAR &HEA00
110 IF MEM$(&HEA00,3)<>HEXCHR$("C3 1E EA") LOADM "RS.OBJ"
120 DEFUSR0=&HEA00       :'OPEN    :STR
130 DEFUSR1=&HEA00+3     :'CLOSE   : *
140 DEFUSR2=&HEA00+6     :'LINPUT  :STR
150 DEFUSR3=&HEA00+9     :'INPUTN  :STR
160 DEFUSR4=&HEA00+12    :'PRINTS  :STR
170 DEFUSR5=&HEA00+15    :'PRINT   :STR
180 DEFUSR6=&HEA00+18    :'LOC     :INT
190 DEFUSR7=&HEA00+21    :'EOF     :INT
200 DEFUSR8=&HEA00+24    :'PGETC1  :INT
210 DEFUSR9=&HEA00+27    :'PPUTC1  :INT
220 '
230 A$=USR0("O6N83XNCCNZ")         :'OPEN
240 A$=USR1("")                    :'CLOSE
250 C=10:A$=USR2(STRING$(C,".")):'LINPUT
260 C=10:A$=USR3(CHR$(C)+STRING$(C,".")):'INPUTN
270 A$=USR4("STRINGS")             :'SEND STRING
280 A$=USR5("ONE LINE")            :'SEND 1 LINE
290 A=USR6(0)                      :'GET BUF SIZE
300 A=USR7(0)                      :'EOF ?
310 A=USR8(0)                      :'GET 1 BYTE
320 A=USR9(0)                      :'PUT 1 BYTE
```

リスト 7-4 簡単なターミナルプログラム(100～210 行はリスト 7-3 と同じ)

```
100 CLEAR &HEA00
110 IF MEM$(&HEA00,3)<>HEXCHR$("C3 1E EA") THEN LOADM "RS.OBJ"
120 DEFUSR0=&HEA00       :'OPEN    :STR
130 DEFUSR1=&HEA00+3     :'CLOSE   : *
140 DEFUSR2=&HEA00+6     :'LINPUT  :STR
```

```
150 DEFUSR3=&HEA00+9     :'INPUTN :STR
160 DEFUSR4=&HEA00+12    :'PRINTS :STR
170 DEFUSR5=&HEA00+15    :'PRINT  :STR
180 DEFUSR6=&HEA00+18    :'LOC    :INT
190 DEFUSR7=&HEA00+21    :'EOF    :INT
200 DEFUSR8=&HEA00+24    :'PGETC1 :INT
210 DEFUSR9=&HEA00+27    :'PPUTC1 :INT
220 '100 - 210 ｷﾞｮｳ ﾊ ﾘｽﾄ3 ﾉ ﾓﾉｦ ﾂｶｳﾉﾃﾞｱﾙ
230 A$=USR0("C6N83XNCCNZ")       :'OPEN
240 L$=""
250 IF USR7(0) THEN PRINT "GET END":END 'EOF
260 A=USR6(0):IF A=0 THEN 300           'LOC > 0?
270 A$=USR3(CHR$(1,0))
280 A$=RIGHT$(A$,1):A=ASC(A$)           'GET AND DISPLAY
290 IF (A>=&H20) OR (A=13) THEN PRINT A$;:GOTO 260
300 C$=INKEY$(0)                        'READ KEY
310 IF (C$=CHR$(0)) OR (L$=C$) THEN 340 'YUUKOU ?
320 IF C$=CHR$(13) THEN A$=USR5(""):GOTO 340
330 A$=USR4(C$)                         'TX IT
340 L$=C$:GOTO 250
```

ここでこの章のプログラムの**手抜きなどについて**書いておくことにする。

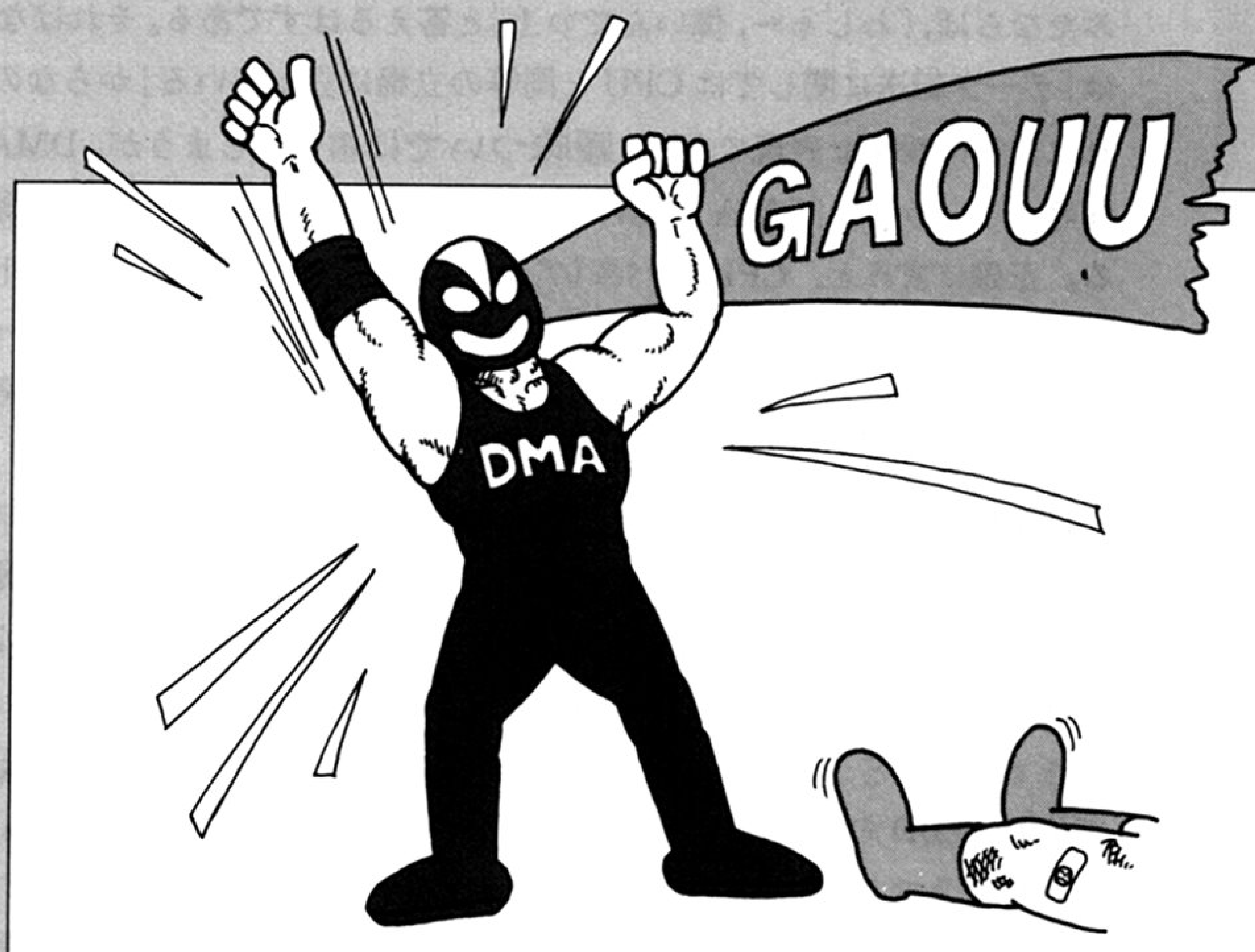
まずはXON, XOFF制御についてである。やってってだんだん面倒になってきたので,「送られてきたXON/XOFFはバッファにたまる」ようになっている。ただしバッファのトップがXONもしくはXOFFであったならば,「こちらからデータを送るときにチェックして捨てる」ようにしている。もしも向こうから「XON/XOFFと他のコードも混ぜて送ってきたら」受信に混ざりこむことになる。必要ならば表示ルーチンなどで取り除いていただきたい。これは,バイナリデータとして,たまたまXON/XOFF ($11_H/13_H$) が送られてくる場合もあるのでこうしたのだが,本当はどうすべきだったのであろうか。またCLOSE時の動作もよく分かんないのである。turbo BASICでは「O」でオープンしたときだけエンドコードを送るようになっているようであるが,「C」では送らないようである。またその逆のエンドコードを受け取ったときの動作も不明瞭である。turbo BASICでは,LINPUTがエンドコードを受け取ったときは1行が終了したとみなしてリターン,INPUT $で入力文字数を指定した場合はエンドコードが来たとしても,指定文字数までリターンしない,などとなっている。ここらへんの細かな動作は,BASICのマニュアルにきっちり書いてあるわけではないので,**なんだかよくわかんない**のである。

おっとここで思い出した。この章のプログラムでは,「EOFはこれから読み出そうとしているデータがエンドコードだったら−1を返す」ようになっている。よって不幸にしてINPUT $などでその「エンドコードを読み出してしまったりすると」EOFでなくなってしまうのである。注意していただきたい。しかしLINPUTでエンドコードは読み出さないようになっているから,エンドコードが一度来たら,それ以降はずっとEOFである。

なんだかんだ書いたが,はっきり言って結局はUSR8, USR9の「生データ読み出し,書き込み」が最終兵器となるような気がする。つらつら考えるに,この二つの機能＋LOC関数＋OPENコマンド＋あぶったイカさえあれば,ほかには何もいらないよーな気がしてしまうのである。というわけで,本当にやるのならば大いに改造していただきたいと思うしだいである。

# 第8章

# DMA




DMAはヘビー級である

# 第8章
# DMAはヘビー級である・・・・・・・・

この章では，Z80 ファミリーの中でも特に「**荒技使い**」との呼び声が高い Z80 DMA についてやるのである。残念ながらこの石は turbo にしか付いていないが，それはそれで仕方がないのである。ちなみにこの石をノン turbo の X1 に付けようとする場合は，本体基板に対する改造が必要となる。CTC や SIO のようにボードに載せて I/O スロットにさすということはできないのである。

DMA というのは「Direct Memory Access」の頭文字を取って作られた言葉である。その点からすれば，DMA 動作をする LSI は DMAC（DMA Controller）と呼ばれるべきなのだが，Zilog 社はアルファベット 3 文字にしたかったようである。それはさておき，この石はデータ転送を主要な業務とする LSI である。M は Memory の M であるが，これは別にメインメモリとは限らず，I/O でもよいのである。この点，誤解のないよーに。

さて，Z80 ファミリーには PIO，CTC，SIO などがあるが，DMA はこの中でも CPU と互角の地位にあり，他の周辺 LSI とは一線を画しているのである。もしも DMA に聞いてみたならば，「わしゃー，偉いんでい！」と答えるはずである。それはなぜかというと，DMA は「データ転送に関しては CPU と同等の立場に立っている」からなのである。う〜ん，これは非常に曖昧な表現である。**曖昧ついでに**書いてしまうが，DMA がどれだけ CPU と同等かというと，「DMA が動いている間は，CPU が止まらざるを得ない」ほどなのである。正確に言うと，CPU はバス（アドレスバス，データバス，コントロールバスの三種の神器）を DMA に明け渡してしまうのである。しかし，ここではハードウェアがどーのこーのという話はあまりしないことにする。詳しく勉強したい人は参考文献 4 である。

まず何よりも DMA が何をするかについて説明しておく。

DMA にできることといえば，後にも先にも，データを CPU より高速に転送/サーチできるということ以外にはない（CPU と同じか，より遅いのならば LDIR なり OUTI なりでやってしまえばよい）。この「速い」ということは非常に大事なことなのである。これが一番はっきりと表面に出てくるのは，8 インチドライブや，5 インチの 2HD を使おうとしたときである。どういうことかというと，これらのディスクのデータ転送速度はやたらと速くて（2D，2DD の約 2 倍），4MHz の Z80 では追い付かないのである。具体的には，たとえば読む場合である。X1 では，ディスクからデータを読むということは，FDC（フロッピーディスクコントローラ：第 9 章を参照）に適当なコマンドを与えた後に，FDC のステータスを見張っていて，FDC が「データを受け取ってくれ！」と言ってきたら，データを IN 命令で持ってくるということなのであるが，5 インチの倍密度（2D や 2DD がそう）の場合なら Z80 でもゆうゆう処理できるのだが，8 インチ，2HD などになると，お手上げになってしまうのである。人から聞いた話によると 8 インチ FD の場合「Z80 の 4MHz ではギリギリ」なのだそうだ。私は実際に，DMA を持っていない MZ-80B（もちろん CPU は 4MHz

の Z80）に 8 インチドライブをつなごうとして失敗した人を知っている。そういうわけで，DMA が X1 turbo に付いているのである。

以上のことからも分かるように，DMA とフロッピーディスクとは深い関係にある。しかし DMA の偉いところは，決して「フロッピーディスクのリード/ライト専用」になっていないことである。つまり，単純なデータ転送や，データサーチなら，フロッピーディスクに関係なくても高速に行なわせることができるのである。実にオイシイ石なのである。

なお，turbo などで 2HD ディスクを使っている場合は FD の動作がおかしくなることもあり得るので，以下のサンプルプログラムを使う間は，ディスクは 2D モードにしておくことをお勧めする。

# DMAの基礎

DMA の基本動作には次の六つがある。

**① メモリ↔メモリ転送**

これは結局のところブロック転送である。LDIR，LDDR と基本的には同じだが，

1） 一定以上のバイト数を転送するなら DMA の方が速い。

2） ソースアドレスをインクリメントし，ディスティネーションアドレスをデクリメントする「順序逆転」などの変態行為を行なわせることができる。

……の点において味がある。

**② メモリ↔I/O 転送**

つまり，グラフィック VRAM の書き込み，読み出しである。もちろん，テキスト VRAM でもよい。矩形領域（画面上の四角）の中を読み出すなどの高等な機能は当然ないが，知恵を絞れば泉が湧くのである。

**③ I/O↔I/O 転送**

最初に「スクロール」という言葉がひらめいたのなら，貴兄はその筋である。そのほかにも，青画面→赤画面転送など，一体何に使ったらよいのか悩んでしまう用途も考えられる。大容量 RAM ボード（EMM）も I/O 空間につながっているので，その方向でも使う価値がある。

**④ メモリサーチ**

大きなデータ中のサーチは結構時間がかかってしまうものであるが，かといって DMA を使うのは大袈裟な気もする。サーチするデータは 1 バイトだけであるがマスクをかけることもできる。たとえば，

&H1?0??1??（?は無視する）

というパターンをサーチできる。特殊な形式のデータを扱う際には役に立つかも。

**⑤ I/O サーチ**

基本的には，やはりグラフィック VRAM のサーチであろうか？ しかし「0 でないデータを捜せ」などはできないのでいまいちのようである。となればテキスト VRAM のサーチということになるが，たかだか 2K バイトをサーチするために DMA を使うのは気がひけ

る。こうなったらEMMの中でもサーチするしかない。

**⑥　転送とサーチの合わせ技**

①〜②の転送とサーチを組み合わせることが可能なのである。つまり，転送しながらあるデータを見つけたら止まれ，というようなことができる。う〜ん何に使おう。

以上が基礎教養である。私はとんでもないことに，この六つの基本動作すべてについてサンプルプログラムを書こうとしているのである。DMAという石がどれだけポップでコーフンするものなのかを示したいともくろんでいるのである。

# DMAは爆発である

では，DMAを使うということが，どんな雰囲気であるのかを簡単に示しておく。

**リスト8-1**は普通の機械語プログラムで，画面をすべて一つの文字で埋めるプログラムである。やっぱり，速すぎるので256回実行している。

リスト8-1　DMAを使わずに

```
                                        .Z80
                                        .PHASE   0E00H
                                ;
07D0                            SIZE    EQU      80*25
                                ;
0E00    26 00                   START:  LD       H,0       ;counter1
0E02    11 07D0                 LOOP1:  LD       DE,SIZE   ;counter2
0E05    01 3000                         LD       BC,3000H
0E08    ED 61                   LOOP2:  OUT      (C),H
0E0A    03                              INC      BC
0E0B    1B                              DEC      DE
0E0C    7A                              LD       A,D
0E0D    B3                              OR       E         ;DE=0 ?
0E0E    C2 0E08                         JP       NZ,LOOP2
                                ;
0E11    24                              INC      H
0E12    C2 0E02                         JP       NZ,LOOP1
                                ;
0E15    C9                              RET
                                ;
                                        END
```

それに対して，**リスト8-2**はDMAを使ってみたものである。分かりやすいように特別なテクニックは使っていない。これも同じく256回実行している。本当はDMAでは似たことを続けて繰り返す場合にはもう少し速くなるのだが，そこまではやっていない。

リスト8-2　DMAしてみたプログラム

```
                                .Z80
                                .PHASE   0E020H
                        ;
07D0                    SIZE    EQU      80*25
                        ;
E020    3E 00                   LD       A,0       ;XOR A...COUNTER
E022    F5              LOOP3:  PUSH     AF
E023    32 E05F                 LD       (DATAAD),A
                        ;
E026    CD E035         MEIOSI: CALL     RESDMA    ;RESET DMA
E029    21 E050                 LD       HL,DMAD   ;LOAD TOP OF COMMANDS
E02C    CD E043                 CALL     WDMA      ;SEND COMMANDS
```

```
                                 ;
E02F    F1                                POP     AF       ;POP COUNTER
E030    3C                                INC     A        ;0 TO 255
E031    C2 E022                           JP      NZ,LOOP3
E034    C9                                RET
                                 ;
E035    21 E03C                  RESDMA:  LD      HL,RESDAT
E038    CD E043                           CALL    WDMA
E03B    C9                                RET
                                 ;
E03C    06                       RESDAT:  DEFB    6        ;NUMBER OF COMMAND
E03D    C3 C3 C3                          DEFB    0C3H,0C3H,0C3H
E040    C3 C3 C3                          DEFB    0C3H,0C3H,0C3H
                                 ;
                                 ;WRITE DMA
                                 ;HL=TOP OF COMMANDS
                                 ;
E043    56                       WDMA:    LD      D,(HL)
E044    01 1F80                           LD      BC,1F80H         ;DMA PORT
E047    23                                INC     HL
E048    04                       REGWR:   INC     B
E049    ED A3                             OUTI
E04B    15                                DEC     D
E04C    C2 E048                           JP      NZ,REGWR
E04F    C9                                RET
                                 ;
E050    0E                       DMAD:    DEFB    0EH      ;NUMBER OF COM.
E051    7D                                DEFB    7DH      ;(D7=0)AND(D1D0<>00)=WR0
E052    E05F                     S_TOP:   DEFW    DATAAD   ;SUB0,1=PORT A TOP ADDR.
E054    07CF                     LEN:     DEFW    SIZE-1   ;LENGTH
E056    24                       S_KIND:  DEFB    24H
                                 ;00100100B        :0??? ?100 = WR1 (PORT A MODE)
E057    18                       D_KIND:  DEFB    18H
                                 ;00011000B        :0??? ?000 = WR2 (PORT B MODE)
                                 ;                          SKIP WR3
E058    CD                                DEFB    0CDH
                                 ;11001101B        :1?????01 = WR4 (WR0 FOR PORT B)
E059    3000                     D_TOP:   DEFW    3000H    ;PORT B TOP ADDR
E05B    9A                                DEFB    9AH      ;WR5 READY=HIGH
E05C    CF                                DEFB    0CFH     ;WR6 (LOAD)
E05D    B3                                DEFB    0B3H     ;WR6 (FORCE READY)LOAD)
E05E    87                                DEFB    087H     ;WR6 (ENABLE DMA)
                                 ;
E05F    00                       DATAAD:  DEFB    0        ;DUMMY
                                 ;
                                          END
```

実行して試してみるには**リスト 8-3** である。最初に普通の機械語で組んだものを実行する。何かキーを押せば，BEEP 音がして画面がくるくると書き換わる。CLS の直後は全画面のアトリビュートが縦 2 倍モードになっていることに注意。次にキーを押せば，DMA 版が走る。明らかに速いことが分かるだろう。最後に同じプログラムを BASIC で組んだものも入れておいた。**すばらしく遅い**ことは言うまでもない。

**リスト 8-3　普通の機械語と DMA を速度比較**

```
100 CLS4
110 DEFINT A-Z
120 CLEAR &HDEEE
130 MEM$(&HE000,16)=HEXCHR$("26 00 11 D0 07 01 00 30 ED 61 03 1B 7A B3 C2 08")
140 MEM$(&HE010, 6)=HEXCHR$("E0 24 C2 02 E0 C9")
150 MEM$(&HE020,16)=HEXCHR$("3E 00 F5 32 5F E0 CD 35 E0 21 50 E0 CD 43 E0 F1")
160 MEM$(&HE030,16)=HEXCHR$("3C C2 22 E0 C9 21 3C E0 CD 43 E0 C9 06 C3 C3 C3")
170 MEM$(&HE040,16)=HEXCHR$("C3 C3 C3 56 01 80 1F 23 04 ED A3 15 C2 48 E0 C9")
180 MEM$(&HE050,16)=HEXCHR$("0E 7D 5F E0 D0 07 24 18 CD 00 30 9A CF B3 87 00")
190 PRINT"NORMAL":GOSUB"SOMEKEY":BEEP:CALL &HE000:BEEP
200 PRINT"DMA"   :GOSUB"SOMEKEY":BEEP:CALL &HE020:BEEP
210 PRINT"BASIC" :GOSUB"SOMEKEY":BEEP
220 FORI=0TO255
230 FORA=&H3000TO&H3000+25*80-1:OUTA,I:NEXT
240 NEXT:BEEP
250 END
```

```
260 LABEL"SOMEKEY"
270 IF INKEY$=""GOTO270
280 RETURN
```

リスト8-2を見ると分かるだろうが，DMAを動かすには1F8＊$_H$番地にコマンドをOUTしてやればよい。逆にDMAからデータを受け取ることも可能で，そのときはIN命令を使う。アドレス中の＊は何でもかまわないが，プログラム中では気分よく1F80$_H$になっている。

ところで，それじゃDMAに与えることのできるコマンドはたった1バイトだけなのか？　という疑問が起こるはずである。もちろんそんなことはない。コマンドの種類は，大きく分けて7種類あり，それぞれのコマンドは「1バイト中のこのビットが立っていたら，この種類」というふうに決まっているのである。簡単に言うなら，CPUとDMAの間であらかじめ決まったフォーマットで8ビットパラレルの通信を行なうようなものである。

## DMAは賢いのである

ではそういう意味も含めて，**リスト8-4**である。これはオールBASICで書いてある。要は，DMAにコマンドを送ってやればよいのであるから，別に機械語を使わなくてもよいのである。唯一の問題がDMAを使う最大の利点＝スピードがなくなってしまうことである。ここではサンプルだからこれでよいのだ。

**リスト8-4　オールBASICでDMAを使ってみる**

```
100 CLEAR&HDFFF
110 POKE&HE000,ASC("X"):'画面を埋める文字
120 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
130 DMA$=RESET$+HEXCHR$("7D 00 E0 CF 07 24 18 CD 00 30 9A CF B3 87")
140 GOSUB"SETDMA"
150 END
160 '
170 LABEL"SETDMA"
180   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
190 RETURN
```

ではプログラムの説明に移る。このプログラムは，全画面に「X」を表示するものである。まず最初にCLEAR命令で領域を確保している。プログラムでは&HE000から&HEFFFまでの4Kバイトを確保しているが，本当に必要なのはたった1バイトなのであった。なぜ1バイトが必要かというと，このプログラムでやっているメモリ→I/O転送ではソース（転送元）を**アドレスで指定**する必要があるためなのだ。プログラムでは，そのソースは&HE000に割り当ててあり，そこにASC（〝X〞）をPOKEしている。

次がRESET$＝〜となって，&HC3が6個並んでいる。これはDMAに送るリセットコマンドである。DMAを初期化するにはとにかく6個のリセットコマンドを送ってやれば，DMAがどんな状態になっていても成功する。これは最悪の場合であって，本当なら6個も送る必要はない。しかし安全のために6個送っている。

次にDMA$＝RESET$＋～となっている部分が，主役のコマンドである。その後，GOSUB〝SETDMA〞で，DMA$に入っている&HC3～&H87までの20個の命令をDMAに送っている。

そこで肩慣らしにリスト8-4に出ているDMAのコマンドの解説を行なう。

**① 概要**

DMAへのコマンドは，大きく分けて7種類ある。それぞれ，WR0～WR6と呼ばれている。WRとはライトレジスタ(Write Register)という意味で，早い話，DMAへコマンドを送るというのは，この七つのレジスタに値を書き込むこと，と解釈すればよい。なお，それぞれのWRの下にはサブレジスタというものがあり，多少複雑になっているが，別に難しいものではない。それぞれのコマンドの分類の仕方だけを**図8-1**に示す。全部で7種類なのだから左端（もしくは右端）の3ビットで判別できるようにすればいいようなものだが，きっとそこにはZilogなりの複雑な事情というものが存在するのであろう。さらに細かな意味や内容を知りたい場合は**図8-2～8-9**を見ていただきたい。

**② $C3_H$＝&H11000011**

図8-1と比べれば，WR6であることが分かるだろう。意味は前述のようにRESETである。

**③ $7D_H$＝&B01111101**

MSB（左端）が0で，右端の二つが00でないからWR0である。そこで図8-2を見る。$D_1$，$D_0$はデータの処理形式を指定する。ここでは「01」だからトランスファー(転送)である。$D_2$（＝1）はデータの転送方向である。DMAは二つのポート（AとB）を持っ

図8-1 DMAライトレジスタ分類早見表

| | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|
| WR0 | 0 | ? | ? | ? | ? | ? | | |
| WR1 | 0 | ? | ? | ? | ? | 1 | 0 | 0 |
| WR2 | 0 | ? | ? | ? | ? | 0 | 0 | 0 |
| WR3 | 1 | ? | ? | ? | ? | ? | 0 | 0 |
| WR4 | 1 | | | ? | ? | ? | 0 | 1 |
| WR5 | 1 | 0 | ? | ? | ? | 0 | 1 | 0 |
| WR6 | 1 | ? | ? | ? | ? | ? | 1 | 1 |

WR0: $D_1$，$D_0$ → 00でない

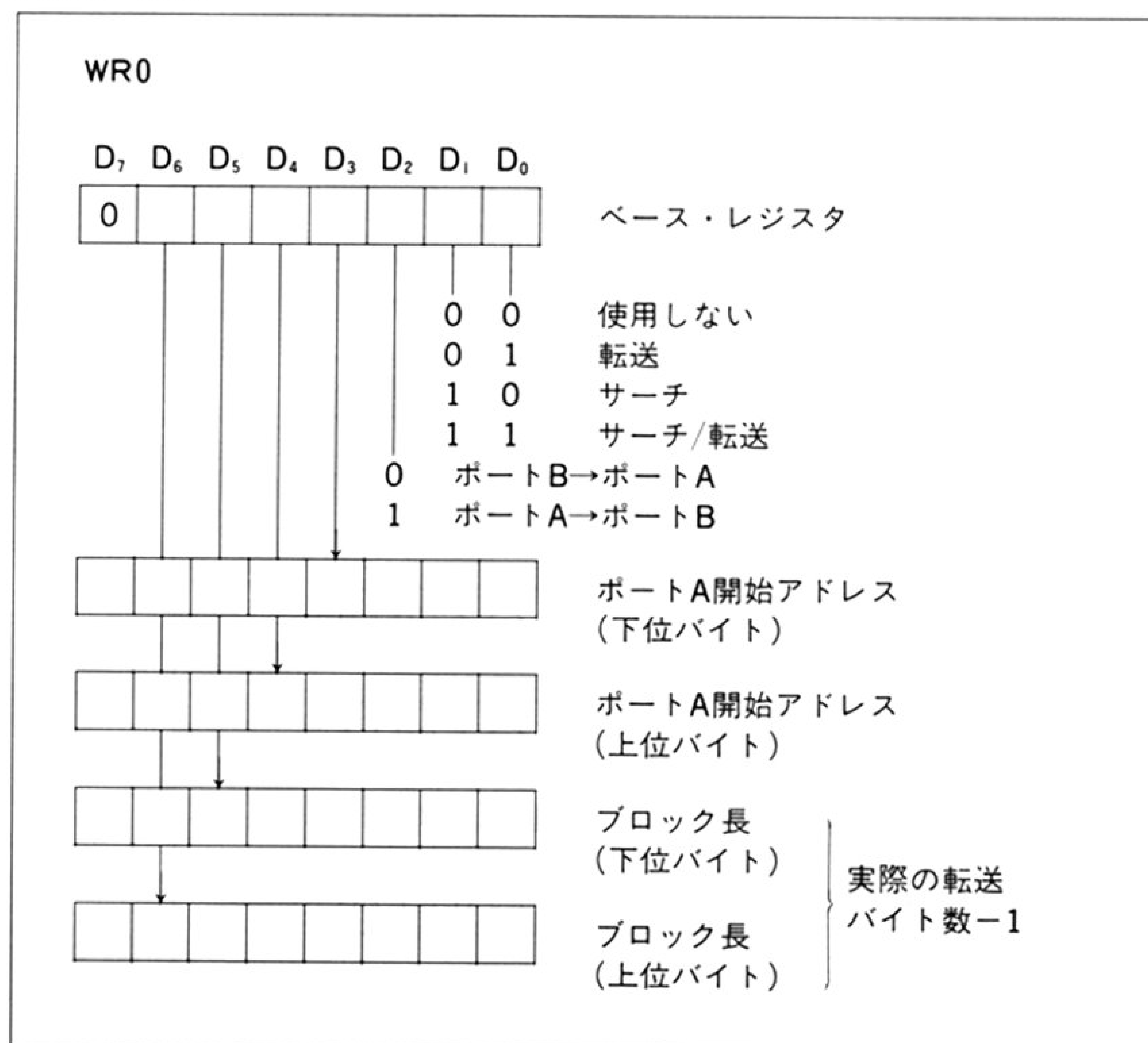
WR4: $D_6$，$D_5$ → 11はダメ

ていて，ソース（転送元）とディスティネーション（転送先）に割り当てるのである。AとBの間には，別にどうという違いはないので，迷う必要はない。ここではA→Bの方向にしている。$D_6$，$D_5$，$D_4$，$D_3$はポートAの開始アドレスと，ブロックレングス（処理バイト数－1）を設定するかどうかのフラグになっている（Z80 DMAの用語ではポインタビットという）。1であれば新しく設定するということになる。0の場合は，設定しないということであり，その場合は以前に設定されたものが使われることになる。図8-2からも分かるように，それぞれを上位，下位に分けて設定するようになっていて，新しく設定するとなれば，LSB（右の方）から指定した順に$1F80_H$にOUTしてやればよい。ここでは4ビットとも1であるから，全部指定することになる。つまり，$00_H$，$E0_H$，$CF_H$，$07_H$が，それぞれサブレジスタの0～3に設定される。処理バイト数が$07CF_H$＝80×25－1であることに注意。1引いておくのだ。念のために言うが，もしブロックレングスの下位を設定しない場合は，$2D_H$＝&HB01011101の後に，$00_H$，$E0_H$，$07_H$とすればよい。

**④　$24_H$＝&B00100100**

図8-1よりWR1である。$D_3$により，ポートAをメモリにするか，I/Oにするかを決定する（ポートBはWR2で決める）。$D_5$，$D_4$は1バイト転送した後，ポートAのアドレスをどう変えるかを指定するものである。図8-3にあるように，－1，＋1，固定（変化させない）の三つの方式を指定できる。ここでは，テキストVRAMを1種類の文字で埋めるのであるから，ソースであるポートAのアドレスは$E000_H$に固定することになる。10，11どちらでもよいが，縁起物だから10にしてある。次に$D_6$であるが，これはポートAへのア

図8-2　WR0



クセスのタイミングを変える場合に1にする。その場合，次に1バイトを送ってやればサブレジスタに値を設定できる。ところがどっこい，turboではこの部分は使えないのである。使えたなら，さらに転送速度を上げることができるのであるが仕方がない。

**⑤ $18_H$＝&B00011000**

WR2である。これは指定の対象がポートBになった点を除いてWR1と同じ使い方である。テキストVRAMへ書き込むのだから，ポートBはI/Oで，インクリメントにしてある。

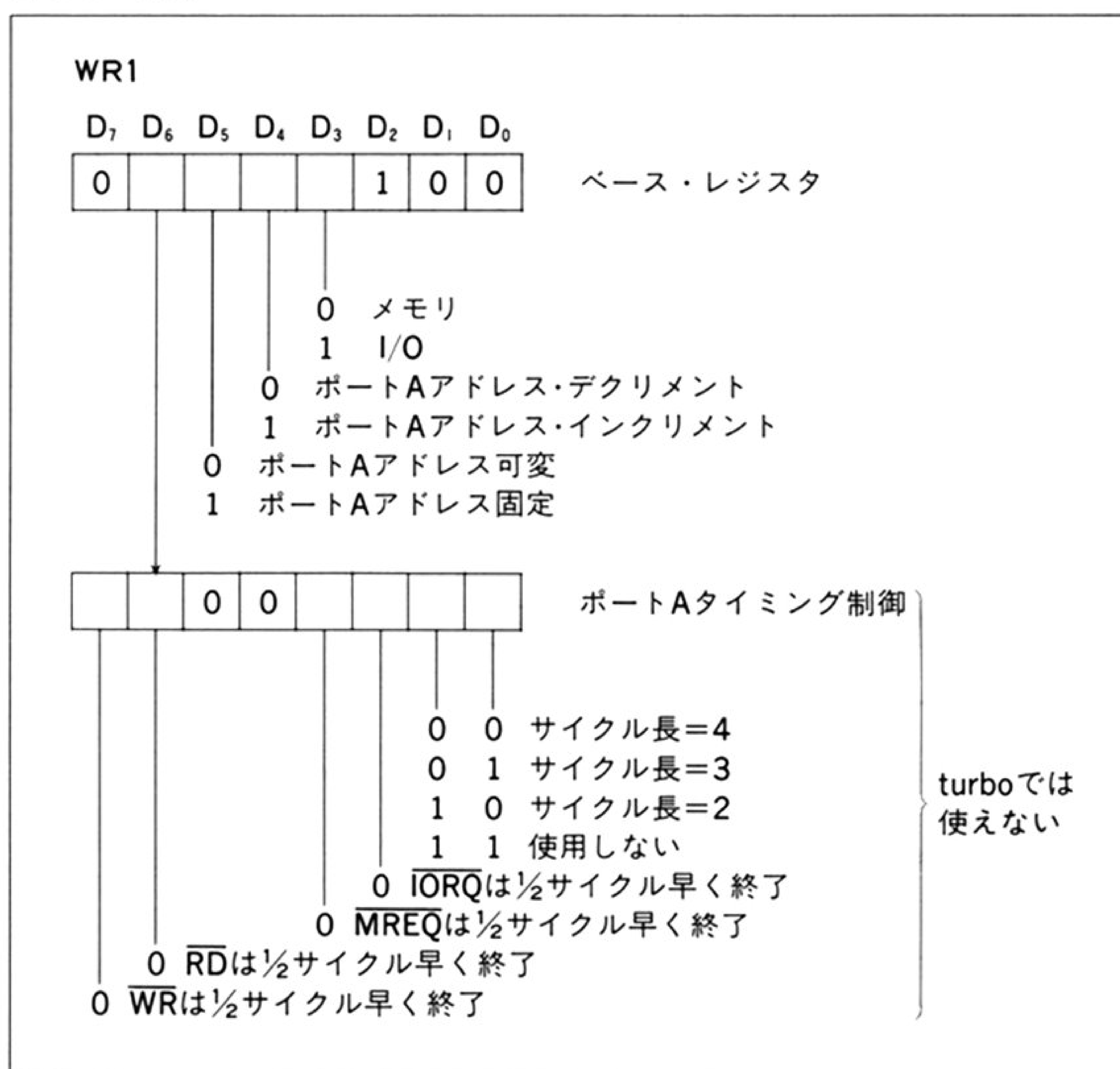
**⑥ $CD_H$＝&B11001110**

WR4である。WR3はサーチ用なので，ここでは抜かしてある。$D_3$,$D_2$がポートB開始アドレスの指定。ここでは，$00_H$，$30_H$がそうである。$D_6$,$D_5$が転送モードの指定である。それぞれの転送モードの詳細は次のとおり。

**バイトモード**：DMAは1バイト処理するたびにCPUにバスを返す。しかし，WR0で指定したブロックレングスだけのバイト数を処理し終わってなく，かつレディ信号（次にWR5で説明する）が有効であれば，その直後に再びバスを要求するので，CPUは1マシンサイクルしか命令を実行できない。早い話がDMAとCPUは交代でバスを使うことになる。turboではDMAを使ってフロッピーディスクにアクセスする場合，このモードを使っている（第9章を参照のこと）。

**バーストモード**：DMAはレディ信号が有効である間はCPUにバスを返さず，プログラム

図8-3　WR1



された処理が終わるまで動作を行ない続ける。レディ信号が無効になった場合はCPUにバスを返すが，再び有効になると，またバスを要求する。この章のサンプルはすべてこのモードを使っている。

**連続(コンティニュアス)モード**：これはDMAが我儘を通すモードで，DMAはプログラミングされたとおりの処理を終わるまで，レディ信号が無効であってもCPUにバスを返さない。つまりCPUはDMAが処理を終えるまで，何もできないことになる。特に素早い応答が要求されている場合に，このモードが使われる。

$D_4$は割り込み制御バイトと呼ばれるサブレジスタを指すものである。サンプルでは使っ

図8-4　WR2

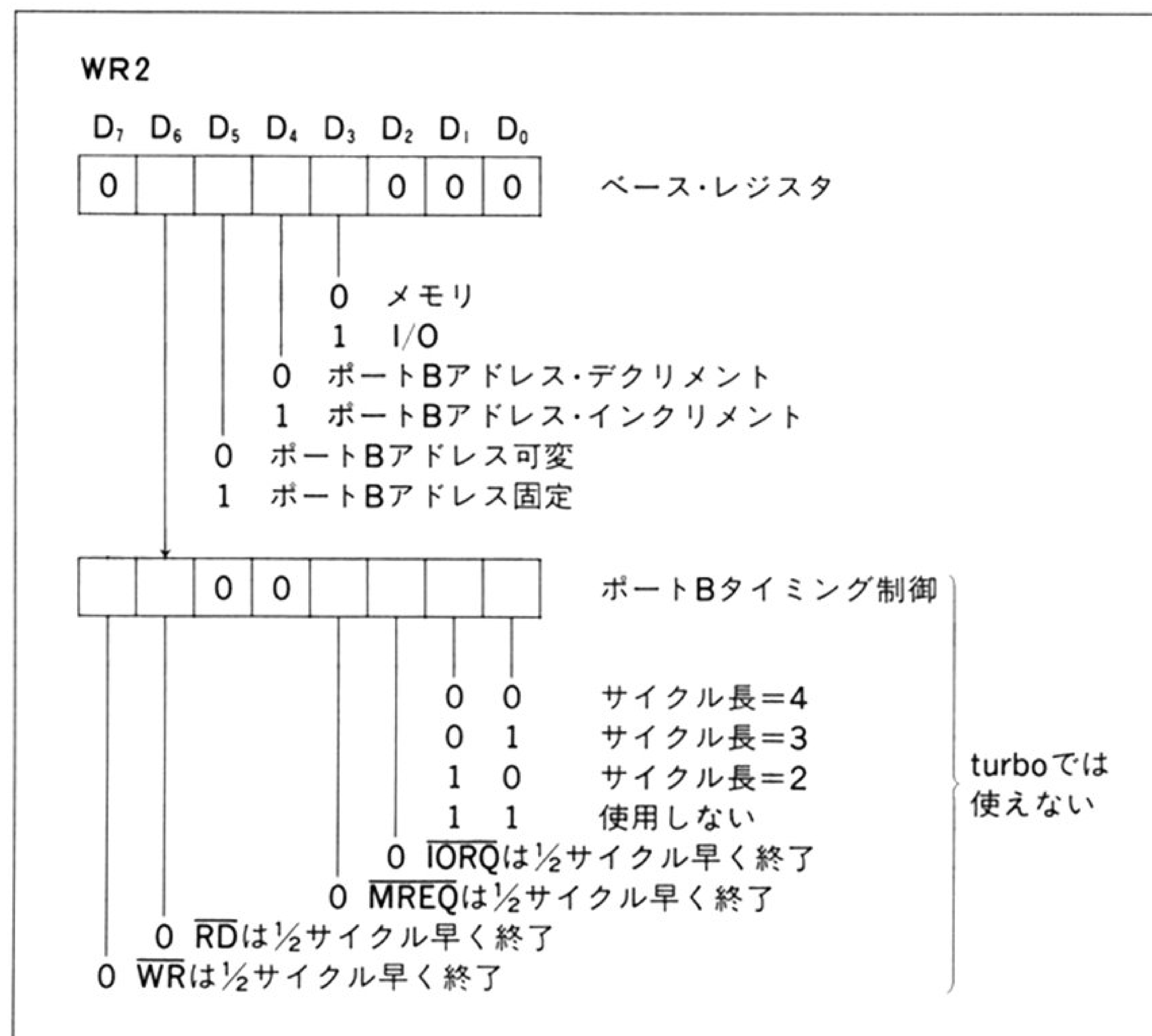
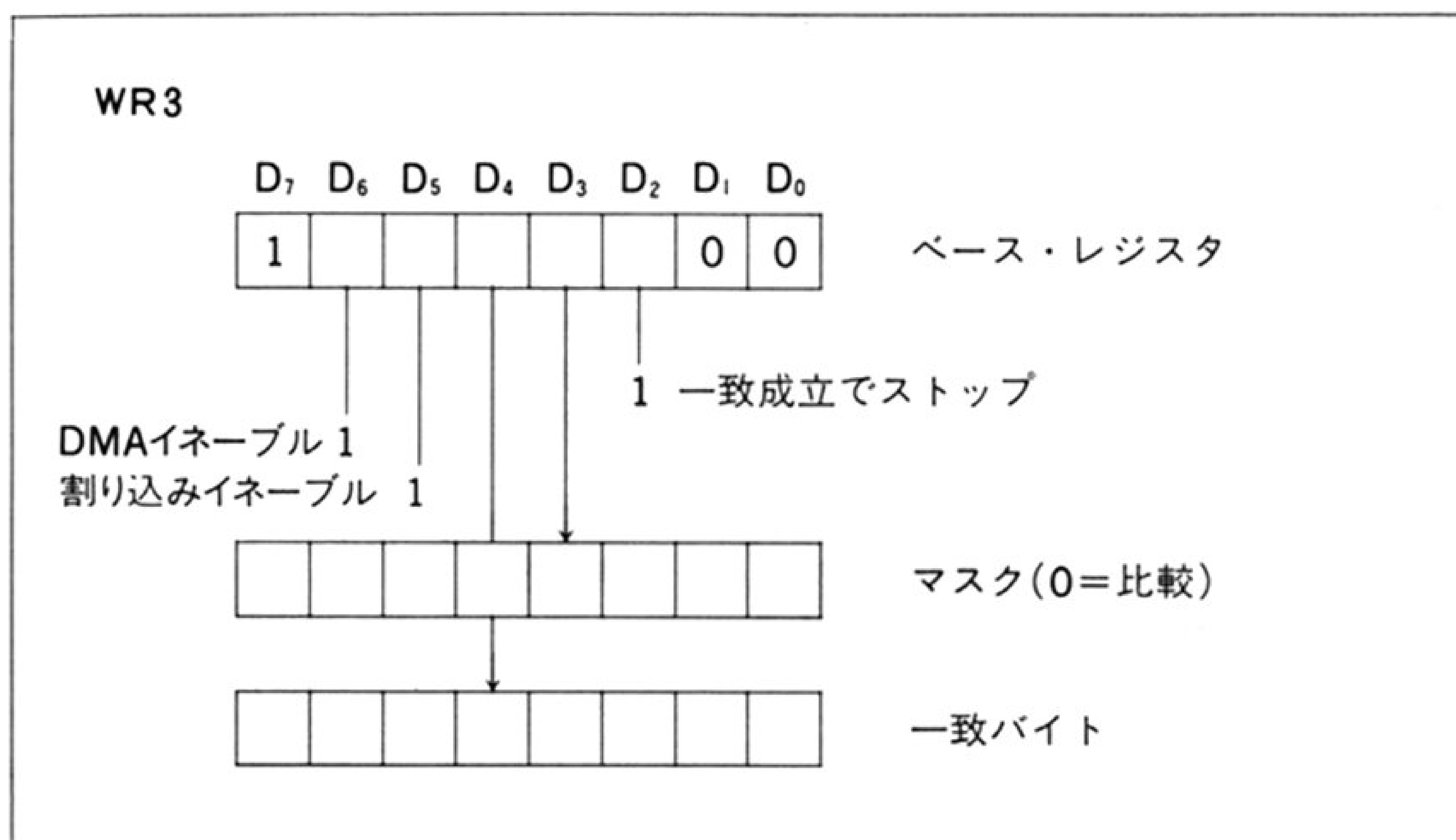



図8-5　WR3



ていないが一応説明しておく。まず，パルス発生(ビット2)はturboでは使えない。これは外部デバイスにパルスを送るものであるから，turboにとっては必要なものではない。

次にインタラプト・オン（マッチ，エンド・オブ・ブロック，レディ）の三つであるが，これはそれぞれの条件のときに割り込みを起こせということ。ステータス・アフェクツ・ベクトルというのは割り込みベクトルを，割り込み要因ごとに変化させるかどうかのフラグである。パルス制御バイトは，パルス発生を使えないのであるからturboでは意味がない。割り込みベクトルはDMAからCPUに割り込みをかけるときのベクトルの下位アドレスを指定するもの。モード2の割り込みをきちんと理解してからいじるように（割

図8-6　WR4

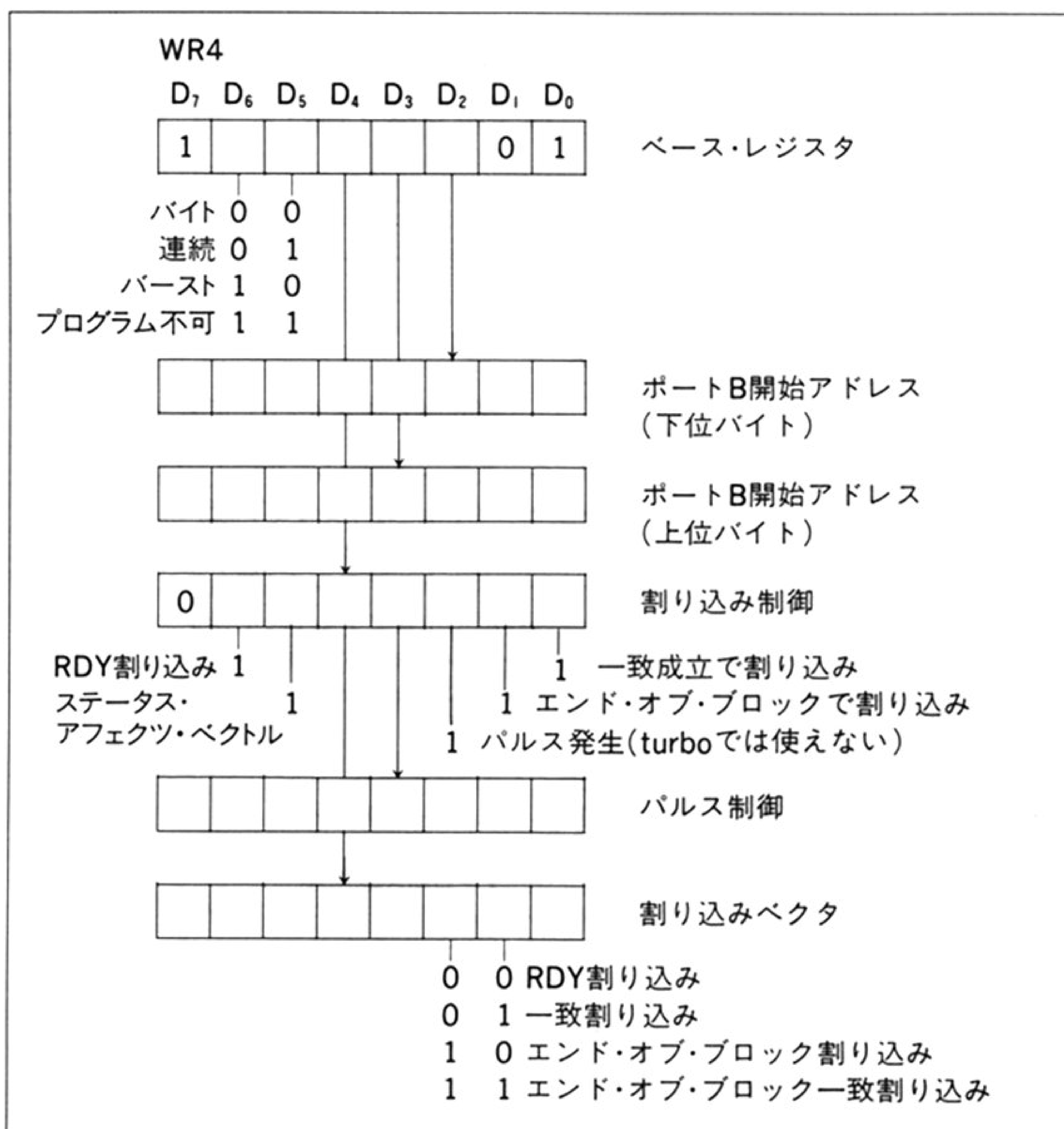
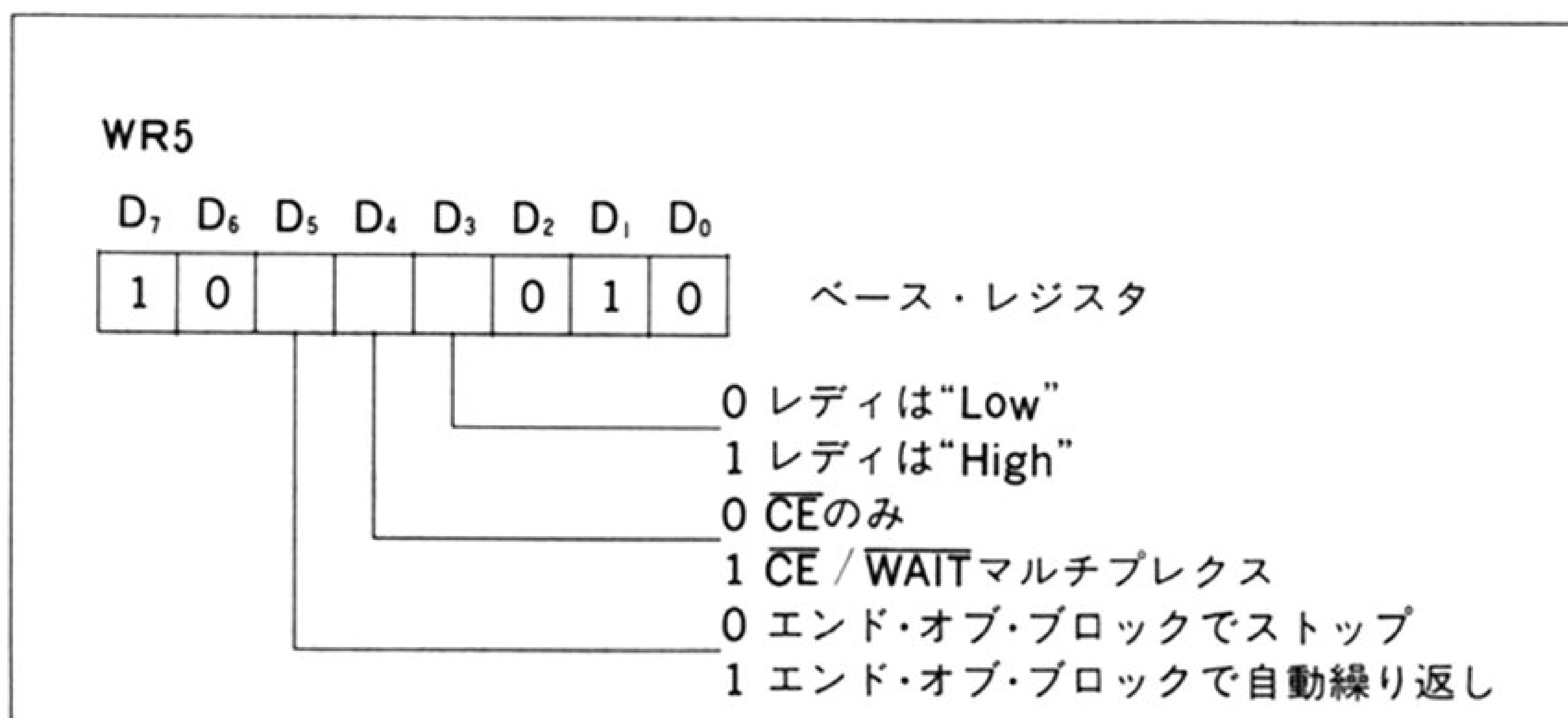



図8-7　WR5



り込みに関しては第4章を参照のこと)。

⑦ **$9A_H$＝&B10011010**

WR5である。まず$D_3$であるが，DMAにはRDYという端子があって，そこへ入ってくる信号をレディ信号という。$D_3$はレディ信号がHighのとき有効にするか，Lowのときに有効にするかの指定である。turboでは通常はこの信号はHighになっているので，このビットは1にしておく。逆に，FDCからデータ要求が来るとLowになるので，ディスクの読み書きをする場合は0にして，Lowを有効とする。$D_4$は$\overline{CE}/\overline{WAIT}$の使い方である。turboではこのビットを1にして$\overline{WAIT}$を受け付けるようにすること。$D_5$はオートリスタートの指定だが，これは0にして「エンド・オブ・ブロックで終決」にする。つまり，ブロックレングス分の処理を終えたらバスをCPUに返して止まるということである。もしオートリスタートにしたなら，レディ信号が有効である限り再スタートするので，止まってくれないことになる。

図8-8 リードレジスタの内容

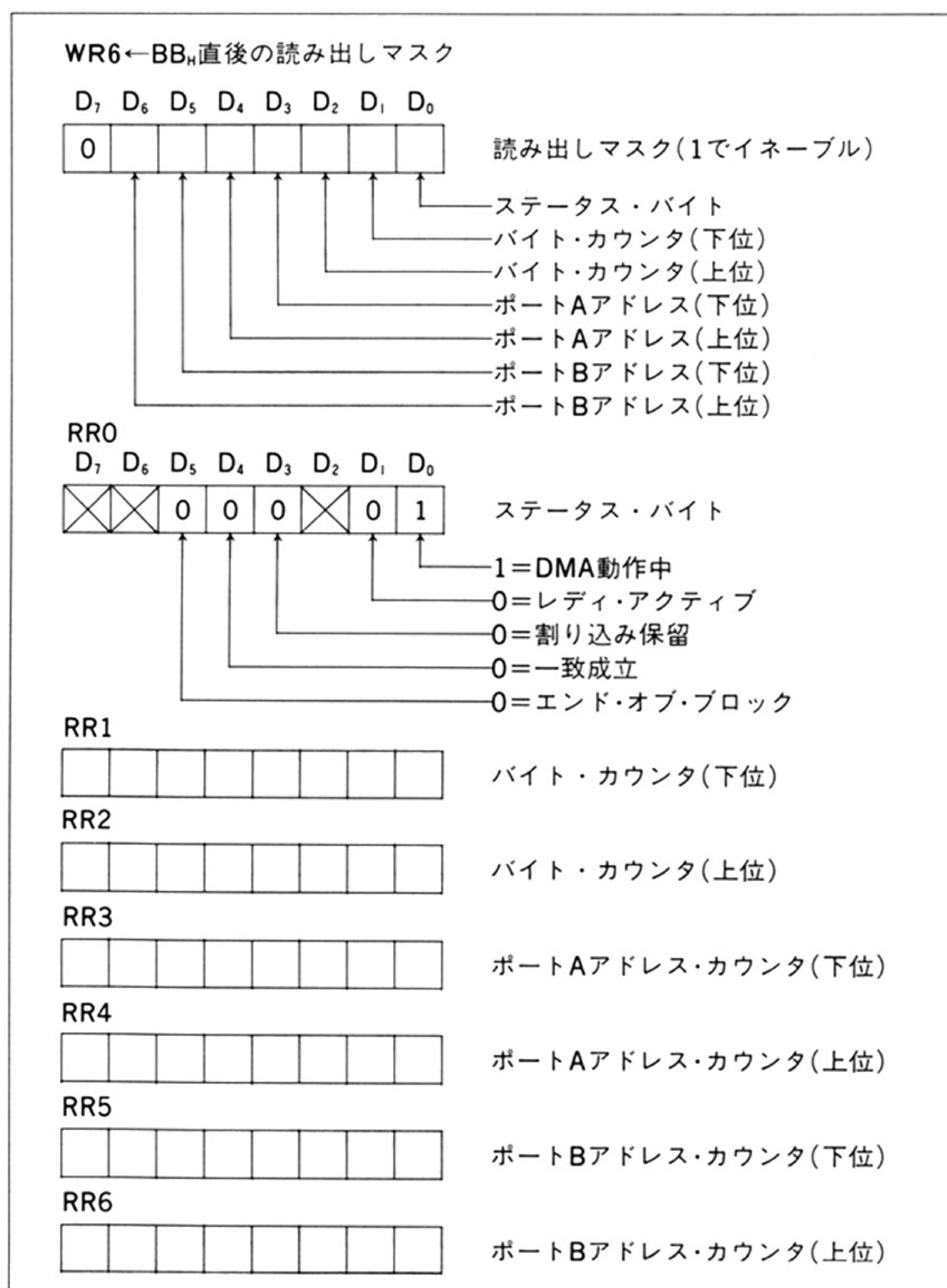


図 8-9　WR6

**WR6**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | 1 | 1 |

| | |
|---|---|
| 83$_H$ | ディセーブルコマンド。87$_H$(イネーブル)の逆。 |
| 87$_H$ | イネーブルコマンド。このコマンドを受け取ると，DMAは直ちにバス要求を出し動作を開始する。このコマンド以外のコマンドは，すべてDMAをディセーブルする。 |
| 8B$_H$ | ステータス・バイト初期再設定コマンド。ステータス・バイトのビット4(一致成立)および5(エンド・オブ・ブロック)を初期化する。 |
| A3$_H$ | 割り込みリセットおよびディセーブルコマンド。Z80を使っているturboには関係ない。8080などがRETIの代わりに使う。 |
| A7$_H$ | 読み出しシーケンス起動コマンド。このコマンドが送られると，マスク設定コマンド(BB$_H$)で指定されたレジスタを順番に読み出せる。リードレジスタの図解を参照。 |
| AB$_H$ | 割り込みイネーブルコマンド。電源投入時，DMAは割り込みディセーブル状態で起動するので，DMAの割り込み機能を使用するときはこのコマンドを送る。このコマンドをセットした後，割り込み条件が発生すると，DMAはCPUに対して割り込み要求を出し続けるので，割り込み処理ルーチンのはじめの方で割り込みディセーブルコマンド(AF$_H$)を送ること。割り込み処理終了後，ふたたび割り込みイネーブルコマンドを送る前に，割り込み条件をリセットするためにステータスバイト初期再設定コマンド(8B$_H$)を送る必要がある。 |
| AF$_H$ | 割り込みディセーブルコマンド。DMA の割り込み処理ルーチンのはじめの方にこのコマンドが必要。そのルーチンの終わりには，ステータス・バイト初期再設定コマンド(8B$_H$)と割り込みイネーブルコマンド(AB$_H$)が必要。 |
| B3$_H$ | 強制RDYコマンド。このコマンドによって，RDY信号は強制的に常にアクティブであるとみなされる。このコマンドは，ロードコマンド(CF$_H$)によってリセットされるので，ロードコマンドより後に送ってやる必要がある。また，バイトモードでは動作しない。 |
| B7$_H$ | RETI後イネーブルコマンド。インタラプト・オン・レディ・モードのときのみ使用される(WR4，割り込み制御バイトのビット6)。このモードでRDY信号がアクティブになるとDMAは割り込み要求を出す。割り込み処理後，DMAにバス要求を出させたい場合，RETI後イネーブルコマンド(B7$_H$)，DMAイネーブルコマンド(87$_H$)を送った後，RETI命令を実行すればよい。 |
| BB$_H$ | 読み出しマスク設定コマンド。このコマンドに続く1バイトデータは，読み出しマスクレジスタへセットされる。マスクでセットされた読み出しレジスタの読み出しシーケンスは，コマンドA7$_H$によって起動する。 |
| BF$_H$ | ステータス・バイト読み出しコマンド。ステータス・バイト(1バイト)だけを読む場合のコマンド。 |
| C3$_H$ | リセット・コマンド。電源投入時やDMAのプログラム打ち切り時に使用される。 |
| C7$_H$ | WR1のポートA可変タイミングを標準Z80タイミングにリセットする(コマンドC3$_H$にもこの機能がある。ただし，turboでは可変タイミングは使えない)。 |
| CB$_H$ | WR2のポートB可変タイミングのリセット(同上)。 |
| CF$_H$ | ロードコマンド。ソース，ディスティネーションのアドレスレジスタの内容をアドレスカウンタにロードする。同時に，バイトカウンタのクリアも行なう。このコマンドによって，ソース側のアドレスは直ちにカウンタにロードされるが，ディスティネーション側のアドレスは，最初のインクリメントorデクリメントの際にロードされるのでディスティネーション・アドレスが固定の場合は，特別な操作を必要とする。また，このコマンドは，強制RDYコマンド(B3$_H$)およびステータス・バイトのビット0(DMA動作中)をリセットしてしまう。 |
| D3$_H$ | コンティニューコマンド。バイトカウンタをクリアするが，両ポートのアドレスカウンタはロードせずに現状の値を保持する。 |

**⑧ $CF_H$＝&B11001111**

WR6である。WR6はビットごとに意味を持つわけではない。$CF_H$はロードコマンドであるが，これは設定された値をワーク用のカウンタにロードするものである。早い話が，「よーい，どん！」の「よーい」である。

**⑨ $B3_H$＝&B10110011**

強制レディコマンド。つまり，RDY信号を常に有効と見なせ，というコマンド。WR5の$9A_H$でHighを有効にしているから，本当は必要ない。しかし，RDY信号が関係なくて，とにかくブロックレングス分の処理をさせたい場合には入れておく方がよい。「よーい，どん！」の「よーい」と「どん！」の間に，「途中で止まるな」と耳元にささやくよーなものである。

**⑩ $87_H$＝&B10000111**

WR6で，イネーブルコマンド。要するに「どん！」で，DMAはこのコマンドを受け取ると，それっとばかりに動き出す。DMAが動き出すと，バスの管理はCPUを離れてDMAに移る。こうなるとCPUは命令を読み出すこと（つまりフェッチ）もできない。すなわち金縛りにあうわけである。だからZ80のプログラムとしては，続けて何事もなかったかのように「転送直後にすること」が書かれていればよいことになる。

以上がこのサンプルの解説である。はっきり言って異常に面倒臭いが，それは最初のうちだけで，一度コマンド列を作ってしまえば，やることを変えるのに，それほど手間はかからない。

そこで，基本的なバリエーションを示してしまえばよい，というわけで，突如としてサンプルプログラムが山津波を起こすのであった。

**A） リスト8-5** （正しい手抜き法）

DMAでは，設定し直さなければ，以前に設定したライトレジスタとサブレジスタを使うことになる。つまり，似たような処理を続けてDMAに行なわせる場合，当然レジスタの設定も似てくるから，いちいち再設定しなくてもよいレジスタが出てくる。リスト8-5は，そこ極端な例で，WR6以外は同じままでよいので，150〜190行の間のループではソースへASCIIコードをPOKEした後，$CF_H$＝ロード，$B3_H$＝強制レディ，$87_H$＝イネーブルだけをDMAに送っている。変化しているのはソースアドレスではなく，ソースアドレスの内容であるからWR0のポートA開始アドレスは再設定する必要がない。やっていることは，リスト8-3と同じく，0〜255のASCIIコードの文字を画面中に書いているだけである。ループの中ではDMAに送るのは3バイト（本当は$B3_H$も抜かしてよい）だけになったので，結構速くなっている。

**リスト8-5　正しい手抜き法**

```
100 CLEAR&HDFFF
110 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3")
120 DMA$=RESET$+HEXCHR$("7D 00 E0 D0 07 24 18 CD 00 30 9A")
130 GOSUB"SETDMA"
```

```
140 '
150 FORK=0TO255
160   POKE&HE000,K:'アスキーコードをソース・アドレスへPOKE。
170   DMA$=HEXCHR$("CF B3 87"):'←これだけですよ、これだけ。
180   GOSUB"SETDMA"
190 NEXT
200 END
210 '
220 LABEL"SETDMA"
230   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
240 RETURN
```

### B) リスト 8-6 (メモリ→メモリ)

単純なブロック転送である。たかだか 64 バイトの転送なら，MEM$(〜)=MEM$(〜) で充分なのであるが，結局サンプルとはそういうものなのである。40 行と 90 行でモニタに飛んでメモリをダンプしている。それ以外に特筆することはない。

リスト 8-6　メモリ→メモリのブロック転送

```
10 CLEAR &HDFFF:CLS4
20 MEM$(&HE000,&H90)=STRING$(&H90,0)  :'メモリをクリア
30 MEM$(&HE000,&H40)=STRING$(&H40,"!"):'ソースをセット
40 KEY0,"DE000 E08F"+CHR$(13)+"R"+CHR$(13):MON
50 A$=INKEY$(1)
60 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット  データ
70 DMA$=RESET$+HEXCHR$("7D 00 E0 3F 00 14 10 CD 40 E0 9A CF B3 87")
80 GOSUB"SETDMA"
90 KEY0,"DE000 E08F"+CHR$(13)+"R"+CHR$(13):MON
100 END
110 '
120 LABEL"SETDMA"
130   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
140 RETURN
```

### C) リスト 8-7 (メモリ→I/O)

「CLS 4」と大体同じことをするプログラムである。具体的には，テキスト VRAM へ 32(スペースの ASCII コード)，アトリビュートへ 7，漢字 VRAM とグラフィック VRAM へ 0 を書き込む。ただしグラフィックの方はバンク 0 か 1，どちらか一方，アクセス可能な状態になっているバンクだけをクリアする。気になったので 100 回実行して「CLS 4」と速度を比べてみた。約 2 倍の速度であった。140 行の HEXCHR$の中は機械語ルーチンである。ソースリストは**リスト 8-8** である。OUTI を使ったテクニックで少しでも速くしようということをもくろんでいる（本当は JR を JP にするともっと速くなる)。USR0 (DMA$) を実行すると，DE レジスタに DMA$の内容へのポインタ，B レジスタに文字列の長さが入って $E000_H$にサブルーチンコールするのである。また，230 行の 0，32，7，は転送するデータなので，0→2 などのように変えると 5 分間ぐらいは楽しめるであろう (100 回ループは消すんだよ)。

リスト 8-7　「CLS 4」モドキ

```
100 DEFINT A-Z
110 INIT:KLIST0:CONSOLE0,25
```

```
120 CLEAR&HDFFF
130 DEFUSR0=&HE000
140 MEM$(&HE000,12)=HEXCHR$("78 EB 01 80 1F 04 ED A3 3D 20 FA C9")
150 '機械語ルーチンを作って速くしたのだ。
160 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
170 DMA$=RESET$+HEXCHR$("7D 20 E0 CF C7 24 18 CD 00 38 9A CF 87")
180 '漢字ＶＲＡＭとグラフィックＶＲＡＭをクリア。
190 DMA$=DMA$+  HEXCHR$("4D 21        07        C9    30    CF 87")
200 'テキストＶＲＡＭを３２（スペース）に。
210 DMA$=DMA$+  HEXCHR$("0D 22                  C9    28    CF 87")
220 'アトリビュートを７（白）に。
230 POKE&HE020,0,32,7:'それぞれのデータです。
240 TIME=0:FORK=1TO100:D$=USR0(DMA$):NEXT:T1=TIME
250 TIME=0:FORK=1TO100:CLS4:NEXT:T2=TIME
260 PRINTT1,T2
270 END
```

リスト 8-8　高速版"SETDMA"

```
                                    .Z80
                                    .PHASE   0E000H  ;OR ANY PLACE
                            ;
                            ;DE=ADDRESS TO DATA,B=COUNTER
E000     78                         LD       A,B
E001     EB                         EX       DE,HL            ;HL POINTS DATA
E002     01 1F80                    LD       BC,1F80H         ;DMA ADDRESS
E005     04                 LOOP:   INC      B
E006     ED A3                      OUTI
E008     3D                         DEC      A
E009     20 FA                      JR       NZ,LOOP
                            ;
E00B     C9                         RET
                            ;
                                    END
```

### D）リスト 8-9（I/O → I/O）

DMA のありがたみは，やはり LDIR が使えず，なおかつポインタとして BC レジスタしか使えない I/O へのアクセス時に七色するのである。最初はテキスト VRAM →テキスト VRAM の転送である。110 行にあるどうしようもない文字列を 0 行目にプリントした後，同順，逆順に転送する。アトリビュートや漢字 VRAM には触っていないので，何かに使おうとする場合は注意が必要である。3 とおりのコマンド列を作っておいたから，「'」(REM) を消して使っていただきたい。特に 180 行の「同じ行内へ逆順に転送する」というやつは，ソースとディスティネーションが重なるとどうなるかという意味も含んでいる(結果は別にどうということもないが)。

リスト 8-9　I/O→I/O(文字列)転送

```
100 CLS4
110 PRINT"0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmn"
120 PRINTSTRING$(160," ")
130 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
140 '
150 '行を替えて同順
160  DMA$=RESET$+HEXCHR$("7D 00 30 4F 00 1C 18 CD 50 30 9A CF B3 87")
170 '同じ行で逆順
180 'DMA$=RESET$+HEXCHR$("7D 00 30 4F 00 1C 08 CD 4F 30 9A CF B3 87")
190 '行を替えて逆順
200 'DMA$=RESET$+HEXCHR$("7D 00 30 4F 00 1C 08 CD 9F 30 9A CF B3 87")
210 '
220 GOSUB"SETDMA"
```

```
230 LOCATE 0,10
240 END
250 '
260 LABEL"SETDMA"
270   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
280 RETURN
```

### E) リスト 8-10 (I/O → I/O)

人呼んで「回転星」という。このプログラムはEMM（大容量RAMボード）が必要だから，持っていない人は指をくわえて見ていていただきたい。

このプログラムが何をしているかというと，EMM ↔ G-RAMの間でデータ転送をしているのである。グラフィックデータはモノクロで1ページ当たり640×200＝16Kバイトである，EMMの容量は320Kバイトだから，計算すれば320÷16＝20画面分が1枚のEMMに入ってしまう。8色なら320/48＝6.66……であるから6画面分。640×400ならばその半分の3画面分である。

**リスト 8-10　回転星**

```
100 WIDTH80,25:INIT:KLIST0:CLS4
110 OUT&HD00,0:OUT&HD01,0:OUT&HD02,0:'ＥＭＭの内部アドレスを０にする。
120 'データをＥＭＭに送りこむ。（ ２０面分 ）
130 S=360/20/5:FORJ=1TO20
140 POLY(320,100),90,4,144,90+J*S,720+90+J*S:'星形を書く
150 PAINT(0,0),2,4:PAINT(320,100),1,2
160 GOSUB"EMMW":CLS4
170 NEXT:BEEP
180 '
190 INIT:CLS4:F=0:Q=0:B=0
200 OUT&HD00,0:OUT&HD01,0:OUT&HD02,0:'ＥＭＭの内部アドレスを０にする。
210 Q=((Q+1)AND7):B=((Q+1)AND7)
220 FORJ=1TO20
230 IF 0 GOSUB"EMMR" ELSE GOSUB"EMMR-2":'次々とＧＲＡＭへ転送する。
240 NEXT:GOTO 200
250 END
260 '
270 LABEL"EMMW":'ＧＲＡＭからＥＭＭへ転送する。
280 DMA$=HEXCHR$("C3C3C3 C3 C3 C3 7D 03 0D FF 3F 2C 18 CD 00 40 9A CF 01 CF 87")
290 GOSUB"SETDMA"
300 RETURN
310 '
320 LABEL"EMMR":'ＥＭＭからＧＲＡＭへ転送する。
330 DMA$=HEXCHR$("C3 C3 C3 C3 C3 C3 7D 03 0D FF 3F 2C 18 CD 00 40 9A CF B3 87")
340 GOSUB"SETDMA"
350 RETURN
360 '
370 LABEL"EMMR-2":'画面スワッピングも使いながらＥＭＭからＧＲＡＭへ転送する。
380 IF F THEN 420
390 DMA$=HEXCHR$("C3 C3 C3 C3 C3 C3 7D 03 0D FF 3F 2C 18 CD 00 40 9A CF B3 87")
400 GOSUB"SETDMA":PALET@ B,Q,B,Q,B,B,B,B
410 GOTO 440
420 DMA$=HEXCHR$("C3 C3 C3 C3 C3 C3 7D 03 0D FF 3F 2C 18 CD 00 80 9A CF B3 87")
430 GOSUB"SETDMA":PALET@ B,B,Q,Q,B,B,B,B
440 F=(F=0):RETURN
450 '
460 LABEL"SETDMA"
470  FOR I=1 TO LEN(DMA$):OUT &H1F80,ASC(MID$(DMA$,I,1)):NEXT
480 RETURN
```

このプログラムはDMAを使い，EMMからG-RAMへデータを転送することにより，一気にアニメーションしてしまおうというものである。ここでは星を回しているだけだが，どんな図形だろうと同じ手間だから，結構面白いことができるはずである。

というところで，リスト8-10の解説を行なう。130〜170行が星型のグラフィックを描き，EMMへ送り込むルーチンである。EMMの内部アドレスは自動的にインクリメントするから，110行と200行でEMMの内部アドレスを設定するだけで済む。

さて，実はこのサンプルは，

**とっても大事なのである。**

というのは，DMAではディスティネーションが**固定アドレスの場合は特別な操作が必要**なのである。EMMへのデータ書き込みは，固定アドレスへの転送だから，これにひっかかるのである。具体的には，ディスティネーションに使用されるポート（この場合はポートA）に実際にアドレスがロードされるのは，最初の増減（インクリメント/デクリメント）が行なわれたときなのだ。だからアドレス固定の場合は死ぬまでロードされない。そのために，少々手間をかけてやる必要がある。早い話が，とりあえず一度ポートAを**ソースに指定してから**$CF_H$（ロードコマンド）を送ってやるのである。ソースの方は，ロードコマンドが送られると，その時点でちゃんとアドレスがロードされるからOK。その直後に「やっぱりポートAはディスティネーションにするね」と，ひっくり返すのである。280行にあるHEXCHR$の中の後ろの方の「CF 01」がその部分である。$01_H$（WR0：B→A指定）が鍵なのである。その後は「CF 87」で再ロード，DMAイネーブルで走ってくれる。

次に230行の「IF 0～」というやつであるが，これは2とおり用意したEMM→G-RAM転送のうち，どちらを選ぶかの分岐点になっている。GOSUB〝EMMR〟の方がシンプルだが，転送している様子が見えてしまうのであまり美しくない。そこで〝EMMR-2〟では青画面（もちろん色はパレットで変えてある）を表示しつつ，赤画面へ転送し，転送後青画面を引っ込めて，赤画面を表示，次は赤画面と青画面の役割を入れ替えて同じことを繰り返す，ということをやっている。どちらもそれほど速くないが，これはDMAのせいではなく，460行からのサブルーチン〝SETDMA〟がトロいのである。リスト8-8の機械語サブルーチンを使えばほぼ2倍の速度になる。**リスト8-11, 8-12**にDMAを使わずにCPUがシコシコと転送するプログラムも載せておくので，turboユーザーでない人も試して欲しい。ただしNEW BASIC (CZ-8F01 V 2.0) でないとペイントにいらだつと思われるので覚悟するよーに。

なお，リスト8-11でUSR0，USR1に文字列を渡しているのは，&H8000が実数型になってしまうためである。さらに補足すると，リスト8-12のそれぞれの先頭にある「CP 3」はAレジスタが渡された変数のタイプを持っているので，それをチェックしているのである。文字列型以外であれば，タイプミスマッチのエラーが発生するようにしてある。肝心の速度の比較であるが，リスト8-10とたいして変わらないようである。

**リスト8-11　DMAを使わない回転星**

```
100 CLEAR&HDFFF
110 DEFUSR0=&HE000:DEFUSR1=&HE030
120 MEM$(&HE000,16)=HEXCHR$("FE 03 28 04 3E 0D DD E9 EB 46 23 4E 21 00 40 D9")
130 MEM$(&HE010,16)=HEXCHR$("01 03 0D D9 ED 78 03 D9 ED 79 D9 2B 7C B5 C2 14")
140 MEM$(&HE020,2) =HEXCHR$("E0 C9")
150 MEM$(&HE030,16)=HEXCHR$("FE 03 28 04 3E 0D DD E9 EB 46 23 4E D9 21 00 40")
160 MEM$(&HE040,16)=HEXCHR$("01 03 0D ED 78 D9 ED 79 03 D9 2B 7C B5 C2 43 E0")
170 MEM$(&HE050,1) =HEXCHR$("C9")
180 WIDTH80:INIT:CLS4
190 OUT&HD00,0:OUT&HD01,0:OUT&HD02,0:'ＥＭＭの内部アドレスを０にする。
200 'データをＥＭＭに送りこむ。（２０面分 ）
210 S=360/20/5:FORJ=1TO20
220 POLY(320,100),90,4,144,90+J*S,720+90+J*S:'星形を書く
230 PAINT(0,0),2,4:PAINT(320,100),1,2
240 GOSUB"EMMW-NM":CLS4
```

```
250 NEXT:BEEP
260 '
270 INIT:CLS4:F=0:Q=0:B=0
280 OUT&HD00,0:OUT&HD01,0:OUT&HD02,0:'ＥＭＭの内部アドレスを０にする。
290 Q=((Q+1)AND7):B=((Q+1)AND7)
300 FORJ=1TO20
310 IF 0 GOSUB"EMMR-NM" ELSE GOSUB"EMMR-NM-2":'次々とＧＲＡＭへ転送する。
320 NEXT:GOTO 280
330 END
340 '
350 LABEL"EMMW-NM":'ＧＲＡＭからＥＭＭへ転送する。
360 D$=USR0(HEXCHR$("4000"))
370 RETURN
380 '
390 LABEL"EMMR-NM":'ＥＭＭからＧＲＡＭへ転送する。
400 D$=USR1(HEXCHR$("4000"))
410 RETURN
420 '
430 LABEL"EMMR-NM-2":'画面スワッピングも使いながらＥＭＭからＧＲＡＭへ転送する。
440 IF F THEN 480
450 D$=USR1(HEXCHR$("4000"))
460 PALET0,B:PALET1,Q:PALET2,B:PALET3,Q
470 GOTO 500
480 D$=USR1(HEXCHR$("8000"))
490 PALET0,B:PALET1,B:PALET2,Q:PALET3,Q
500 F=(F=0):RETURN
```

リスト 8-12　EMM↔G-RAM サブルーチン

```
                                   .PHASE  0E000H
                                   .Z80
                            ;
                            ;STAR
                            ;
E000    FE 03                      CP      3         ;CHECK VAR.TYPE
E002    28 04                      JR      Z,OKGO1   ;HL POINTS.INT.?
E004    3E 0D                      LD      A,13      ;TYPE MISMATCH
E006    DD E9                      JP      (IX)      ;ERROR ROUTINE
                            ;
E008    EB                  OKGO1: EX      DE,HL
E009    46                         LD      B,(HL)    ;HIGH
E00A    23                         INC     HL
E00B    4E                         LD      C,(HL)    ;LOW
E00C    21 4000                    LD      HL,4000H        ;COUNTER
E00F    D9                         EXX
E010    01 0D03                    LD      BC,0D03H        ;EMM DATA
                            ;
E013    D9                         EXX
E014    ED 78               LOOP1: IN      A,(C)     ;FM GRAM
E016    03                         INC     BC
E017    D9                         EXX
E018    ED 79                      OUT     (C),A
E01A    D9                         EXX
E01B    2B                         DEC     HL
E01C    7C                         LD      A,H
E01D    B5                         OR      L
E01E    C2 E014                    JP      NZ,LOOP1
                            ;
E021    C9                         RET
                            ;
                            ;
E022                               DS      14        ;DASAI
                            ;
E030    FE 03                      CP      3         ;CHECK VAR.TYPE
E032    28 04                      JR      Z,OKGO2   ;HL POINTS INT.?
E034    3E 0D                      LD      A,13      ;TYPE MISMATCH
E036    DD E9                      JP      (IX)      ;ERROR ROUTINE
                            ;
E038    EB                  OKGO2: EX      DE,HL
E039    46                         LD      B,(HL)    ;HIGH
E03A    23                         INC     HL
E03B    4E                         LD      C,(HL)    ;LOW
```

```
E03C   D9                    EXX
E03D   21 4000               LD     HL,4000H      ;COUNTER
E040   01 0D03               LD     BC,0D03H      ;EMM DATA
                      ;
E043   ED 78          LOOP2: IN     A,(C)   ;FM EMM
E045   D9                    EXX
E046   ED 79                 OUT    (C),A
E048   03                    INC    BC
E049   D9                    EXX
E04A   2B                    DEC    HL
E04B   7C                    LD     A,H
E04C   B5                    OR     L
E04D   C2 E043               JP     NZ,LOOP2
                      ;
E050   C9                    RET
                      ;
                             END
```

**F）リスト 8-13, 8-14**（テキストスクロール）

別に言うことはないが，このプログラムを走らせると，画面に表示されている漢字が一瞬化ける。これは三つの転送（テキスト VRAM，アトリビュート，漢字 VRAM）の間に時間差があるため。つまり〝SETDMA〟が遅いからである。

**リスト 8-13　テキストスクロールアップ**

```
10 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
20 'テキストVRAM
30 DMA$=RESET$+HEXCHR$("7D 50 30 7F 07 1C 18 CD 00 30 9A CF B3 87")
40 'アトリビュート
50 DMA$=DMA$  +HEXCHR$("7D 50 28 7F 07 1C 18 CD 00 28 9A CF B3 87")
60 '漢字VRAM
70 DMA$=DMA$  +HEXCHR$("7D 50 38 7F 07 1C 18 CD 00 38 9A CF B3 87")
80 GOSUB"SETDMA"
90 END
100 '
110 LABEL"SETDMA"
120   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
130 RETURN
```

**リスト 8-14　テキストスクロールダウン**

```
10 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
20 'テキストVRAM
30 DMA$=RESET$+HEXCHR$("7D 7F 37 7F 07 0C 08 CD CF 37 9A CF B3 87")
40 'アトリビュート
50 DMA$=DMA$  +HEXCHR$("7D 7F 2F 7F 07 0C 08 CD CF 2F 9A CF B3 87")
60 '漢字VRAM
70 DMA$=DMA$  +HEXCHR$("7D 7F 3F 7F 07 0C 08 CD CF 3F 9A CF B3 87")
80 GOSUB"SETDMA"
90 END
100 '
110 LABEL"SETDMA"
120   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
130 RETURN
```

**G）リスト 8-15**（上8ドットグラフィックスクロール）

これは8ドット分グラフィックを上へスクロールさせるプログラムである。

$4050_H$〜$477F_H$ →$4000_H$〜$472F_H$

$4850_H$〜$4F7F_H$→$4800_H$〜$4F2F_H$

⋮

⋮

⋮

$7850_H$〜$7F7F_H$→$7800_H$〜$7F2F_H$

という具合に８回ブロック転送を実行している。これはX1のグラフィックの座標とアドレスの関係からこのようになっているのである。このプログラムではY座標が192〜199に当たる下の方の８ライン分はそのまま残るので，140行のサークル命令の半径を95ぐらいにするとゴミが出る。

**リスト8-15　上８ドットグラフィックスクロール**

```
100 DEFINT A-Z
110 CLEAR&HDFFF
120 DEFUSR0=&HE000
130 MEM$(&HE000,12)=HEXCHR$("78 EB 01 80 1F 04 ED A3 3D 20 FA C9")
140 INIT:CLS4:CIRCLE(320,100),60,6
150 PAINT(320,100),&H26,6
160 SYMBOL(282,100),"朝です",2,2,7,0,PRESET
170 '
180 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
190 '青画面
200 DMA$=RESET$+HEXCHR$("7D 50 40 7F 07 1C 18 CD 00 40 9A CF 87")
210 DMA$=DMA$  +HEXCHR$("15    48              C9    48 9A CF 87")
220 DMA$=DMA$  +HEXCHR$("15    50              C9    50 9A CF 87")
230 DMA$=DMA$  +HEXCHR$("15    58              C9    58 9A CF 87")
240 DMA$=DMA$  +HEXCHR$("15    60              C9    60 9A CF 87")
250 DMA$=DMA$  +HEXCHR$("15    68              C9    68 9A CF 87")
260 DMA$=DMA$  +HEXCHR$("15    70              C9    70 9A CF 87")
270 DMA$=DMA$  +HEXCHR$("15    78              C9    78 9A CF 87")
280 '赤画面
290 DMA$=DMA$  +HEXCHR$("15    80              C9    80 9A CF 87")
300 DMA$=DMA$  +HEXCHR$("15    88              C9    88 9A CF 87")
310 DMA$=DMA$  +HEXCHR$("15    90              C9    90 9A CF 87")
320 DMA$=DMA$  +HEXCHR$("15    98              C9    98 9A CF 87")
330 DMA$=DMA$  +HEXCHR$("15    A0              C9    A0 9A CF 87")
340 DMA$=DMA$  +HEXCHR$("15    A8              C9    A8 9A CF 87")
350 DMA$=DMA$  +HEXCHR$("15    B0              C9    B0 9A CF 87")
360 DMA$=DMA$  +HEXCHR$("15    B8              C9    B8 9A CF 87")
370 '緑画面
380 DMA$=DMA$  +HEXCHR$("15    C0              C9    C0 9A CF 87")
390 DMA$=DMA$  +HEXCHR$("15    C8              C9    C8 9A CF 87")
400 DMA$=DMA$  +HEXCHR$("15    D0              C9    D0 9A CF 87")
410 DMA$=DMA$  +HEXCHR$("15    D8              C9    D8 9A CF 87")
420 DMA$=DMA$  +HEXCHR$("15    E0              C9    E0 9A CF 87")
430 DMA$=DMA$  +HEXCHR$("15    E8              C9    E8 9A CF 87")
440 DMA$=DMA$  +HEXCHR$("15    F0              C9    F0 9A CF 87")
450 DMA$=DMA$  +HEXCHR$("15    F8              C9    F8 9A CF 87")
460 FORK=0TO20:D$=USR0(DMA$):NEXT
470 END
```

**H）リスト8-16**（左８ドットグラフィックスクロール）

$4001_H$〜$7FFF_H$→$4000_H$〜$7FFE_H$

$8001_H$〜$BFFF_H$→$8000_H$〜$BFFE_H$

$C001_H$〜$FFFF_H$→$C000_H$〜$FFFE_H$

の３回ブロック転送をしている。考えられる限りセコいので，80回（640ドット）左へスクロールすると８ドット分上へ上がってしまう。これについては，普通の機械語でも試し

て速度を比べてみた。270行からがそうである。USR1に「8ドット左へスクロールする回数」を渡せばよい。サンプル中では80を渡してある。機械語部分のソースは**リスト8-17**である。本当はありとあらゆる汚い手を使って高速化してみたかったのだが，読みづらくなるので避けたのである。もっとも，このサンプルのように，BCとB′C′をソースとディスティネーションに割り当てるのが一番速いと思う。

**リスト8-16　左8ドットグラフィックスクロール**

```
100 DEFINT A-Z
110 CLEAR&HDFFF
120 DEFUSR0=&HE000:DEFUSR1=&HE100
130 MEM$(&HE000,12)=HEXCHR$("78 EB 01 80 1F 04 ED A3 3D 20 FA C9")
140 INIT:CLS4:CIRCLE(320,100),60,6
150 PAINT(320,100),&H26,6
160 SYMBOL(266,100),"異常です",2,2,7,0,PRESET
170 '
180 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
190 '青画面
200 DMA$=RESET$+HEXCHR$("7D 01 40 FE 3F 1C 18 CD 00 40 9A CF 87")
210 '赤画面
220 DMA$=DMA$  +HEXCHR$("15    80             C9    80 9A CF 87")
230 '緑画面
240 DMA$=DMA$  +HEXCHR$("15    C0             C9    C0 9A CF 87")
250 TIME=0:FORK=0TO80:D$=USR0(DMA$):NEXT:PRINT"BY DMA    =";TIME
260 '
270 MEM$(&HE100,16)=HEXCHR$("FE 02 28 04 3E 0D DD E9 7E B7 C8 57 01 00 40 CD")
280 MEM$(&HE110,16)=HEXCHR$("23 E1 01 00 80 CD 23 E1 01 00 C0 CD 23 E1 15 C2")
290 MEM$(&HE120,16)=HEXCHR$("0C E1 C9 78 D9 47 0E 01 21 FE 3F ED 78 03 D9 ED")
300 MEM$(&HE130,11)=HEXCHR$("79 03 D9 2B 7C B5 C2 2B E1 D9 C9")
310 TIME=0:D=USR1(80):PRINT"NOT BY DMA=";TIME
```

**リスト8-17　DMAを使わずにグラフィックスクロール**

```
                                   .PHASE  0E100H
                                   .Z80

                          ;GRAPHIC SHIFT TO LEFT
                          ;
E100    FE 02                      CP      2          ;CHECK VAR.TYPE
E102    28 04                      JR      Z,OKGO     ;HL POINTS INT.?
E104    3E 0D                      LD      A,13       ;TYPE MISMATCH
E106    DD E9                      JP      (IX)       ;ERROR ROUTINE
                          ;
E108    7E                OKGO:    LD      A,(HL)     ;0-255
E109    B7                         OR      A
E10A    C8                         RET     Z          ;0 TIME
E10B    57                         LD      D,A        ;COPY TIMES
                          ;
E10C    01 4000           LOOP0:   LD      BC,4000H           ;BLUE
E10F    CD E123                    CALL    SHIFT1
                          ;
E112    01 8000                    LD      BC,8000H           ;RED
E115    CD E123                    CALL    SHIFT1
                          ;
E118    01 C000                    LD      BC,0C000H          ;GREEN
E11B    CD E123                    CALL    SHIFT1
                          ;
E11E    15                         DEC     D
E11F    C2 E10C                    JP      NZ,LOOP0
                          ;
E122    C9                         RET                ;RET TO BASIC
                          ;
                          ;SUBROUTINE       BC=TOP OF PAGE
E123    78                SHIFT1:  LD      A,B
E124    D9                         EXX
E125    47                         LD      B,A
E126    0E 01                      LD      C,01       ;B'C'=SOURCE
E128    21 3FFF                    LD      HL,3FFFH           ;COUNTER
                          ;
E12B    ED 78             LOOP1:   IN      A,(C)      ;GET 1 BYTE
E12D    03                         INC     BC         ;INC POINTER
E12E    D9                         EXX
```

```
E12F    ED 79                    OUT   (C),A    ;COPY 1 BYTE
E131    03                       INC   BC       ;INC POINTER
E132    D9                       EXX
E133    2B                       DEC   HL       ;DEC COUNTER
E134    7C                       LD    A,H
E135    B5                       OR    L        ;CHECK COUNTER
E136    C2 E12B                  JP    NZ,LOOP1
                     ;
E139    D9                       EXX
E13A    C9                       RET            ;RET TO MAIN ROUTINE
                     ;
                                 END
```

## I）リスト 8-18（上１ドットグラフィックスクロール）

律儀に上下をつなげてみた。200回スクロールすると元に戻るわけである。メインメモリの E100$_H$番地から E8CF$_H$までの 07CF$_H$＋１バイトをワークエリア（バッファ）にしていることに注意。このサンプルプログラムのように上下をつなげる場合にはどうしても必要である。本当のところは，１ラインのスクロールならワークエリアは80バイトで済むのだが，そうすると手間がかかり，速度に響いてしまう。

79$_H$はポートB→ポートAへの転送を指定している。

リスト 8-18　上１ドットグラフィックスクロール

```
100 DEFINT A-Z
110 CLEAR&HDFFF
120 DEFUSR0=&HE000
130 MEM$(&HE000,12)=HEXCHR$("78 EB 01 80 1F 04 ED A3 3D 20 FA C9")
140 INIT:KLIST0:CLS4:CIRCLE(320,100),60,6
150 PAINT(320,100),&H62
160 SYMBOL(282,100),"朝です",2,2,7,0,PRESET
170 '
180 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
190 '青画面
200 DMA$=RESET$+HEXCHR$("7D 00 40 CF 07 1C 10 CD 00 E1 9A CF 87"):'TO MEM
210 DMA$=DMA$  +HEXCHR$("7D 00 48 CF 37 1C 18 CD 00 40 9A CF 87"):'IN VRAM
220 DMA$=DMA$  +HEXCHR$("79 00 78 7F 07 1C 10 CD 50 E1 9A CF 87"):'FM MEM
230 DMA$=DMA$  +HEXCHR$("79 80 7F 4F 00 1C 10 CD 00 E1 9A CF 87"):'FM MEM
240 '赤画面
250 DMA$=DMA$+RESET$
260 DMA$=DMA$  +HEXCHR$("7D 00 80 CF 07 1C 10 CD 00 E1 9A CF 87"):'TO MEM
270 DMA$=DMA$  +HEXCHR$("7D 00 88 CF 37 1C 18 CD 00 80 9A CF 87"):'IN VRAM
280 DMA$=DMA$  +HEXCHR$("79 00 B8 7F 07 1C 10 CD 50 E1 9A CF 87"):'FM MEM
290 DMA$=DMA$  +HEXCHR$("79 80 BF 4F 00 1C 10 CD 00 E1 9A CF 87"):'FM MEM
300 '緑画面
310 DMA$=DMA$+RESET$
320 DMA$=DMA$  +HEXCHR$("7D 00 C0 CF 07 1C 10 CD 00 E1 9A CF 87"):'TO MEM
330 DMA$=DMA$  +HEXCHR$("7D 00 C8 CF 37 1C 18 CD 00 C0 9A CF 87"):'IN VRAM
340 DMA$=DMA$  +HEXCHR$("79 00 F8 7F 07 1C 10 CD 50 E1 9A CF 87"):'FM MEM
350 DMA$=DMA$  +HEXCHR$("79 80 FF 4F 00 1C 10 CD 00 E1 9A CF 87"):'FM MEM
360 FORK=1TO200:D$=USR0(DMA$):NEXT
370 END
```

## J）リスト 8-19（上２ドットグラフィックスクロール）

これも律儀に上下をつなげてある。残念ながら実行してよーく見ていると色がズレまくっている。ここらへんがDMAでグラフィックを扱う限界なのかもしれないが，「正しい手抜き法」を使えばある程度の改善はできるはずである。挑戦してみていただきたい。

リスト 8-19　上２ドットグラフィックスクロール

```
100 DEFINT A-Z
110 CLEAR&HCFFF
120 DEFUSR0=&HD000
130 MEM$(&HD000,12)=HEXCHR$("78 EB 01 80 1F 04 ED A3 3D 20 FA C9")
```

```
140 INIT:KLIST0:CLS4:CIRCLE(320,100),60,2
150 PAINT(320,100),2,2
160 SYMBOL(240,100),"赤丸急上昇",2,2,7,0,PSET
170 '
180 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット  データ
190 '青画面
200 DMA$=RESET$+HEXCHR$("7D 00 40 CF 07 1C 10 CD 00 E0 9A CF 87"):'TO MEM
210 DMA$=DMA$  +HEXCHR$("7D 00 48 CF 07       CD 00 E8    CF 87"):'TO MEM
220 DMA$=DMA$  +HEXCHR$("7D 00 50 FF 2F    18 CD 00 40    CF 87"):'IN VRAM
230 DMA$=DMA$  +HEXCHR$("79 00 70 7F 07    10 CD 50 E0    CF 87"):'FM MEM
240 DMA$=DMA$  +HEXCHR$("79 00 78 7F 07       CD 50 E8    CF 87"):'FM MEM
250 DMA$=DMA$  +HEXCHR$("79 80 77 4F 00       CD 00 E0    CF 87"):'FM MEM
260 DMA$=DMA$  +HEXCHR$("79 80 7F 4F 00       CD 00 E8    CF 87"):'FM MEM
270 '赤画面
280 DMA$=DMA$+RESET$
290 DMA$=DMA$  +HEXCHR$("7D 00 80 CF 07 1C 10 CD 00 E0 9A CF 87"):'TO MEM
300 DMA$=DMA$  +HEXCHR$("7D 00 88 CF 07       CD 00 E8    CF 87"):'TO MEM
310 DMA$=DMA$  +HEXCHR$("7D 00 90 FF 2F    18 CD 00 80    CF 87"):'IN VRAM
320 DMA$=DMA$  +HEXCHR$("79 00 B0 7F 07    10 CD 50 E0    CF 87"):'FM MEM
330 DMA$=DMA$  +HEXCHR$("79 00 B8 7F 07       CD 50 E8    CF 87"):'FM MEM
340 DMA$=DMA$  +HEXCHR$("79 80 B7 4F 00       CD 00 E0    CF 87"):'FM MEM
350 DMA$=DMA$  +HEXCHR$("79 80 BF 4F 00       CD 00 E8    CF 87"):'FM MEM
360 '緑画面
370 DMA$=DMA$+RESET$
380 DMA$=DMA$  +HEXCHR$("7D 00 C0 CF 07 1C 10 CD 00 E0 9A CF 87"):'TO MEM
390 DMA$=DMA$  +HEXCHR$("7D 00 C8 CF 07       CD 00 E8    CF 87"):'TO MEM
400 DMA$=DMA$  +HEXCHR$("7D 00 D0 FF 2F    18 CD 00 C0    CF 87"):'IN VRAM
410 DMA$=DMA$  +HEXCHR$("79 00 F0 7F 07    10 CD 50 E0    CF 87"):'FM MEM
420 DMA$=DMA$  +HEXCHR$("79 00 F8 7F 07       CD 50 E8    CF 87"):'FM MEM
430 DMA$=DMA$  +HEXCHR$("79 80 F7 4F 00       CD 00 E0    CF 87"):'FM MEM
440 DMA$=DMA$  +HEXCHR$("79 80 FF 4F 00       CD 00 E8    CF 87"):'FM MEM
450 FORK=1TO100:D$=USR0(DMA$):NEXT
460 END
```

以上が転送であった。DMA の動作には，転送以外にサーチ，サーチ＋転送がある。こちらの使い方も基本的には同じなのだが，多少趣が異なるので，まずはごくごく基本のサンプルを示す。

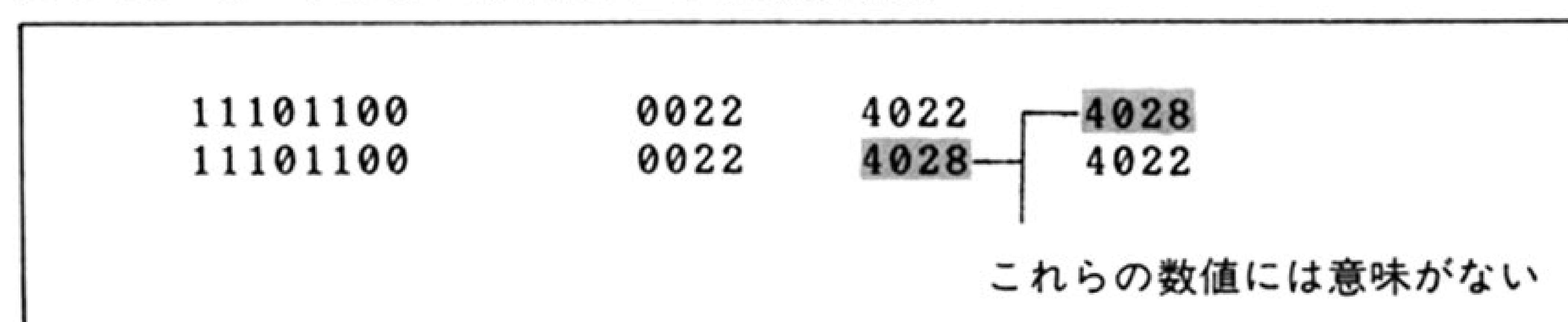
**K）リスト 8-20**

**図 8-10** は実行結果である。まず 1000 行からのサブルーチンを説明する。

リードレジスタについては一応説明してあったが，これは DMA が実行を終了した後で，「どれどれお前さんは何をしたのかね」と言いつつノゾキ見るために付いているのである。特にサーチは実行結果が DMA の内部にしか残らないから，どうしてもここを見る必要がある。そのリードレジスタを DMA から読み出すには何とおりかの方法があるが，別に深く悩まず全部読み出してしまえばよいのであった。まずリスト 8 -20 の 1080 行目からリードレジスタを読み出し，配列 DMA（～）に代入している。

最初に&HBB を OUT しているが，これは WR6 のコマンドで「リード・マスク・フォローズ」（後ろに続くマスクを読め）である。その直後の&H7 がそのマスクである。このマスクの中の対応するビットが立っていれば（1ならば），それに対応する 1 バイトのリードレジスタを読み出すことになる。1110 行のFOR～NEXT 文がそうである。図 8-8を見れ

図 8-10　サーチの基本技（リスト 8-20）実行例



リスト 8-20　サーチの基本技

```
100 INIT:CLS4
110 CLEAR &HDFFF
120 OUT&H4020,&HFF:'マーク
130 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
140 '
150 'ポートAを使って、マスクなしで&B11111111=&HFFをサーチ
160 DMA$=RESET$+HEXCHR$("7E 00 40 30 00 1C    9C 00 FF C1       9A CF 8B 87")
170 GOSUB"DISPDMA"
180 '
190 'ポートBを使って、マスクなしで&B11111111=&HFFをサーチ
200 DMA$=RESET$+HEXCHR$("62       30 00    18 9C 00 FF CD 00 40 9A CF 8B 87")
210 GOSUB"DISPDMA"
220 END
230 '
1000 LABEL"DISPDMA"
1010   GOSUB"SETDMA":GOSUB"READDMA":GOSUB"DMPDMA"
1020 RETURN
1030 '
1040 LABEL"SETDMA"
1050   FORI=1TOLEN(DMA$):OUT&H1F80,ASC(MID$(DMA$,I,1)):NEXT
1060 RETURN
1070 '
1080 LABEL"READDMA"
1090   OUT&H1F80,&HBB
1100   OUT&H1F80,&H7F
1110   FORI=1TO7:DMA(I)=INP(&H1F80):NEXT
1120 RETURN
1130 '
1140 LABEL"DMPDMA"
1150   PRINT RIGHT$("0000000"+BIN$(DMA(1)),8),
1160   FORI=2TO6STEP2:GOSUB"DMPDMA-SUB":NEXT
1170   PRINT
1180 RETURN
1190 '
1200 LABEL"DMPDMA-SUB"
1210   PRINTRIGHT$("000"+HEX$(DMA(I)+DMA(I+1)*256),4),
1220 RETURN
```

ば分かるように，最初の 1 バイト目はステータスバイトと呼ばれ，各ビットごとに意味を持つ。その後の 2 ～ 7 バイト目はそれぞれバイトカウンタ，ポート A アドレス，ポート B アドレスの上下バイトになっている。ただしこれらの値は DMA の構造上から多少大きめの値を示すようになっているので注意が必要である。図 8-10 を見ていただきたい。たしかに，ポート A のアドレスが&H4022 となっている。

それでは 160 行の HEXCHR$の中のコマンドについて解説する。まず，サーチを実行するためには開始アドレスを決め，ポート A，B のどちらかに割り当てなければいけない。A，B どちらでもよいのだが，その場合は WR0 でソース側 (転送元) に指定しておく必要がある。リスト 8-20 ではポート A（160 行）ポート B（200 行）の両方を使った場合を示してある。ポートを取り替えたこと以外は同じことをさせている。さて，中程に $9C_H$ というのがあるが，これが WR3 である。その直後の 00 と FF が WR3 のサブレジスタへの値である。図 8-5 を参照。マスクバイト（00）というのは「ビットが 0 なら比較しろ」という意味を持つ。マッチバイトはサーチすべき 1 バイトのデータである。マスクバイトというのが出てくると，とたんにこんがらがってしまうのだが，

（マッチバイト）OR（マスクバイト）=（ポート A）の指すデータ）OR（マスクバイト）

となれば一致成立ということであると覚えておけばよいだろう。$9C_H$ は $D_2$ が 1 であるからストップ･オン･マッチを指定している。さて，他のコマンドは $8B_H$ 以外は簡単に分かるだろう。$8B_H$ は WR6 のステータスバイト初期再設定コマンドである。理由は不明なのだ

が，サーチを行なう場合はロードコマンド($CF_H$)の後にこのコマンドを送ってやる必要があるみたいである。手元にある資料にはサーチの実例がほとんどなく，また説明も不充分なのである。とにかく，$8B_H$を取り除くと動作がおかしくなる。

**L）リスト 8-21**

この中では，ポートAを使ってサーチしている。最初の170行の例はI/O空間(グラフィックRAM)中の$4021_H$番地にOUTしてある$FF_H$を，$4000_H$からサーチしている。200行は同じように$F0_H$をサーチしている。このデータはないので，ブロックレングス($2F_H$)＋1個のデータをサーチした後に止まる。転送と同じように，ブロックレングスで指定した値＋1が対象になるのである。230行は$FC_H$をサーチしている。これはちょうどサーチ範囲の最後にあるので，ステータスバイト(**図 8-11** の3行目）の$D_5$（＝0でエンド・オブ・ブロック），$D_4$（＝0でデータが一致）の両方とも0になっている。260行はマスクを使ってサーチしている。

　　($FF_H$) OR　($0F_H$)　　　　　　＝$FF_H$

　　(4020の内容＝$FE_H$) OR ($0F_H$) ＝$FF_H$

であるから，これは$4020_H$の$FE_H$で一致している。290行は，マスクバイトが$FF_H$であるから，いかなるデータとでも一致することになる。なぜなら，

　　(××) OR ($FF_H$)＝$FF_H$

　　(△△) OR ($FF_H$)＝$FF_H$

だからである。

320行はサーチはサーチなのだが，WR3へのコマンド中で「見つけたら停止」を指定していない($98_H$)ので，指定したバイト数だけ最後まで処理している。ただしステータスバイトの$D_4$はちゃんと0（発見した）になっている。

次にとても面倒なのだが，読み出したレジスタの値は動作モード，条件などによってかなり複雑に変化する。正しくは参考文献4を見ていただきたいが普通に使う分には「バーストモードのサーチで，データが一致したなら，ソースポートのアドレスは発見アドレスの＋2」と覚えておけばよいだろう。その他の場合は実に変幻自在で，このような状況を押し付けられると，どうも使う気が起きなくなってしまうものである。ただでさえ転送と違って，利用法の少ない動作なのだからなおさらなのである。さらには，もしWR3に「一致不成立時に停止」という動作が指定できたなら，「0以外のデータを見つける」ということもできたはずなのである。実になまぬるい。

**図 8-11　サーチ六態(リスト 8-21)実行例**

```
11101100        0023    4023    4028
00011000        0030    4030    4028
00001000        0030    4030    4028
00101000        0022    4022    4028
11101100        0002    4002    4028
00001000        0030    4030    4028
                        └──────────┘
                        これらの数値には意味がない
```

**リスト 8-21　サーチ六態**

```
100 INIT:CLS4
110 CLEAR &HDFFF
120 OUT&H4020,&HFE
130 OUT&H4021,&HFF
140 OUT&H402F,&HFC
150 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
160 'マスクなしで&B11111111=&HFFをサーチ
170 DMA$=RESET$+HEXCHR$("7E 00 40 2F 00 1C 9C 00 FF C1 9A CF 8B 87")
180 GOSUB"DISPDMA"
190 'マスクなしで&B11110000をサーチ
200 DMA$=RESET$+HEXCHR$("7E 00 40 2F 00 1C 9C 00 F0 C1 9A CF 8B 87")
210 GOSUB"DISPDMA"
220 'マスクなしで&B11111100をサーチ
230 DMA$=RESET$+HEXCHR$("7E 00 40 2F 00 1C 9C 00 FC C1 9A CF 8B 87")
240 GOSUB"DISPDMA"
250 'マスクありで&B1111????をサーチ
260 DMA$=RESET$+HEXCHR$("7E 00 40 2F 00 1C 9C 0F FF C1 9A CF 8B 87")
270 GOSUB"DISPDMA"
280 'マスクありで&B????????をサーチ
290 DMA$=RESET$+HEXCHR$("7E 00 40 2F 00 1C 9C FF FF C1 9A CF 8B 87")
300 GOSUB"DISPDMA"
310 'マスクなしで&B11111111をサーチただし、停止しない
320 DMA$=RESET$+HEXCHR$("7E 00 40 2F 00 1C 98 00 FF C1 9A CF 8B 87")
330 GOSUB"DISPDMA"
340 END
350 '
360 'このあとには、リスト8-20の1000行目以後を入力してください。
```

まあ，文句はこれくらいにして，次には結構使えそうなモードのサーチ＋転送のサンプルを示すのである。

**M）リスト 8-22**

サーチ＋転送の使い方は，WR0 で「転送/サーチ」を指定することと，WR3 によりマスクとマッチバイトを指定する以外は（ロードの後の $8B_H$ もあるけれど）転送と基本的に同じである。もちろん WR3 で「見つけたら停止」を指定しなければ，転送したデータの中に一致するデータがあったということだけが後から分かる。サンプルプログラムの中では「.」（ピリオド）までの文字列をテキスト VRAM へ転送している。もちろんこのとき，ブロックレングスは充分大きな値にしておかなければ，「.」に達する前に動作が停止してしまう。このサンプルを作ってみて，またもや不満が出てしまった。すなわち，「一致する一つ前のデータまで転送」というモードが欲しいのである。

ま，文句ばかり言っても仕方ないか。

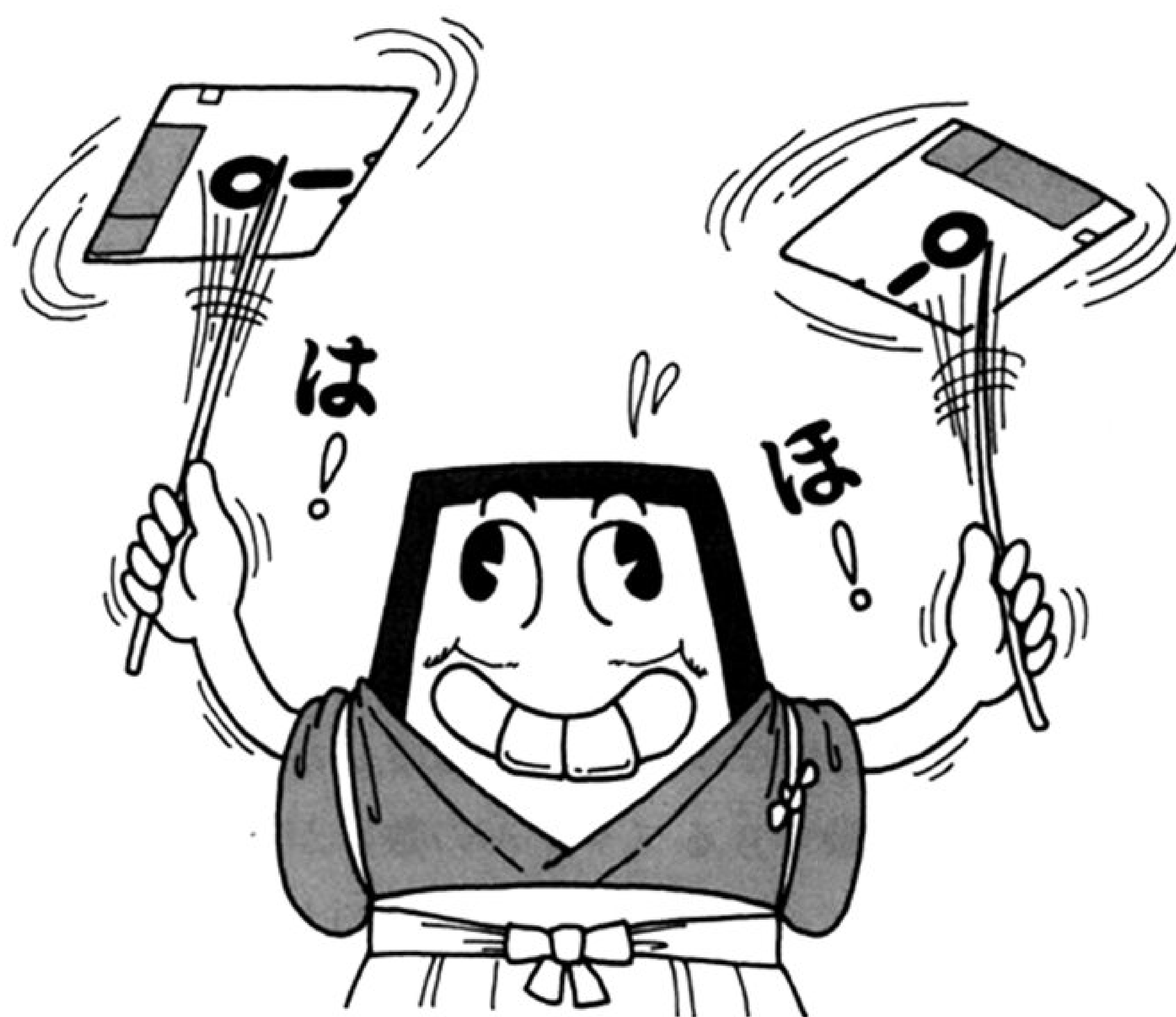
**リスト 8-22　サーチ＋転送**

```
100 INIT:CLS4
110 CLEAR &HDFFF
120 MESSAGE$="WATASI WA DMA DE TENSOU SARERU."
130 MEM$(&HE000,LEN(MESSAGE$))=MESSAGE$:'ソースを用意する
140 LOCATE0,0:PRINTSPACE$(80):'アトリビュートの設定
150 RESET$=HEXCHR$("C3 C3 C3 C3 C3 C3"):'リセット　データ
160 'マスクなしで、"."までを、サーチ＋転送
170 DMA$=RESET$+HEXCHR$("7F 00 E0 00 01 14 18 9C 00 2E CD 00 30 9A CF 8B 87")
180 LOCATE0,3:GOSUB"DISPDMA"
190 END
200 '
210 'このあとには、リスト8-20の1000行目以後を入力してください。
```

以上でDMAの説明は終わる（ほんとは第9章のディスクのところでまたやるけど）。で，DMAの良い点の一つとして挙げることができるのが，BASIC上から機械語を使わずに，機械語と同じ速度，もしくはそれ以上の速度で処理ができるということである。さんざんけなしたけど，サーチだってBASICでやったのなら実にトロいことになってしまうのだ。**とりあえず**DMAにありがとうである。

# 第9章

# フロッピーディスク




ディスクを回すのである

## 第9章

# ディスクを回すのである・・・・・・・

この章はフロッピーディスクである。そこでまずは参考書について一言。

フロッピーディスク関係の本の中で，ストロングタイプのお勧め品は，参考文献10にも挙げてある『最新フロッピ・ディスク装置とその応用ノウハウ』である。この本は1984年の発刊で，残念ながらあんまり「最新」ではなくなってしまったのだが，プロテクト以外のことに関しては，ほとんどすべてが載っているので，是非とも手元に置いておきたい一冊である。

さて，同書によるとフロッピーディスクなるものがこの世に生を受けたのは，1972年のことである。IBMの大型コンピュータに採用されたのだが，確か私の記憶だと，IPL起動時のシステム読み込みに使われていたはずである。このときのものは，8インチの片面単密で，容量は約250Kバイトであった。当時としては，そのように簡単に取り外しができて，しかもランダムアクセス可能な大容量のデバイスは画期的なものだったらしい。

時は移り，それから4年後の1976年に米国シュガート社から5.25インチディスクが発表された(なんでキッチリ5インチにしなかったんだっ。無意味な不満とともに,以下ではしばしば5インチと呼ぶ)。これは当初は容量が80Kバイトしかなかったそうである。

8インチと5インチは，その後着々と世に広がり，また技術の向上に伴って記録密度も上がってくるのだが，三つ子の魂百までと言うように，8インチと5インチはちょっと違う道を歩むことになる。すなわち，巨人IBMの手によって生み出された8インチの方は「IBMフォーマット」という，**誰も逆らえない**スタンダードが根底に確立したのだが，片や5インチの方は，ほとんど野放しに近い状態となり，いろいろなフォーマットや方式が乱立することになった。

その中でも一番の被害者は5インチドライブを持つCP/Mユーザーである。8インチドライブの場合は「片面単密度IBMフォーマット」と一声念じれば，ディレクトリなどに相当するソフト的なフォーマットも含めてすべてが解決したのだが，5インチの場合は，単密度しか読めないやつとか，倍密度しか読めないやつとか，内部のディレクトリのエントリーの数（登録できるファイルの数）が違ったりとか，トラックの数が違ったりとかで，もーむちゃくちゃになってしまったのだ。その状況は今も変わらず，メディアの互換性はほとんど捨ておかれてしまっている。その状況は3.5インチでも同じである。ただ一つ明るい材料といえば，5インチではMS-DOSフォーマットが実質的な標準になりつつあるということだが，それとてもMS-DOSフォーマットへのコンバータが必ず入手できるわけではない。困ったもんだ。

なおこの章のサンプルプログラムは，特にことわっていない限り2Dモードで動くようになっているのでturboZのユーザーの方などは注意されたい。

# フロッピーディスクドライブの種類

今まで「5インチ」,「8インチ」,「2D」,「2DD」,「2HD」と勝手に書いてきたが，ここらでちゃんと説明しておくことにする。

太古の昔には，さまざまな形式のものがあったそうだが，現在の主流はメディアのサイズが，

**8，5.25，3.5，3インチ**

記録方式が，

1S，2S，1D，2D，2DD，2HD

となっている。**表 9-1，9-2** が大まかな分類と大体の容量，個人的な予測である。

表 9-1　メディアの種類

| | 5.25，3.5，3インチ | | 8インチ | |
|---|---|---|---|---|
| | 片面 | 両面 | 片面 | 両面 |
| 単密度 | 1S (80KB) | 2S (160KB) | 1S (250KB) | 2S (500KB) |
| 倍密度 | 1D(160KB) 1DD(320KB) | 2D(320KB) 2DD(640KB) 2HD(1MB) | | 2D (1MB) |

表 9-2　普及度の(個人的な)予測

| サイズ | メジャー度 |
|---|---|
| 8 | ↓ |
| 5.25 | ↗ |
| 3.5 | ↗ |
| 3 | ✝ |

表を見て当然気付くだろうが，記録方式では8インチの方に1Dがない(噂によると，どこぞに1Dの8インチドライブがあるそうだが，多分2Dのドライブの片面だけを使っているのではないかと推測する)。また，2DDや2HDに相当するものもない。理由は簡単である。「**IBMが作らなかった**」からである。それに対して，5，3.5，3インチの方は1Dと2S，1DDと2Dのように，同じ容量なのに違う形式のものがあったりする。この理由も明白である。「IBMが作らなかった」からである。

まずはディスクの回り方などについて説明する。

ディスクの回転速度は8インチが毎分360回転，5インチでは2HDが8インチと同じく毎分360回転，それ以外は300回転である。3.5インチは300回転と600回転の2種類があるが，メディアに記録される信号は同じで互換性はある。3インチは毎分300回転で，

5インチと同じ。回る方向は，5インチで確認したところ，ラベル側から見て右回り（時計回り）であった。8インチはよく見えなかったので不明だが，多分同じであろう。

次にディスクの形状であるが，まずはライトプロテクトの方式について少し触れておく。8インチの場合は，ディスケットの一部を専用の器具を使って切り欠くことによって，ライトプロテクトの状態になる。切り欠いた所に銀紙を張ると元どおりに書き込めるようになる。5インチの場合は逆で，元々切り欠きがあり(最近のゲームソフトなどでは切り欠きのない特別のディスケットを使っているものもあるが)，その状態が書き込み可能である。そこへ銀紙を張るとライトプロテクトされる。だから同じ銀紙でも，8インチの場合と5インチの場合は機能が逆である。

3.5インチの場合は，プラスチック製のスイッチがあり，それをスライドさせることによりライトプロテクトになる。スイッチだから元に戻すのも簡単である。

3インチも3.5インチと同様にプラスチックのスイッチをスライドさせる方式である。ただしスイッチは二つある。この理由は後程述べる。

なお，ライトプロテクトであるが，X1の場合はハード的に書き込みができないようになるが，一部の機種では絶対に書き込めないとは限らず，コントロールプログラムの方でライトプロテクトになっているかどうかをチェックしてやって，なっていたら書き込み動作を中止してやる，という処理が必要なものもある。

次にインデックスホールであるが，8，5，3インチにはあるが，3.5インチにはない。代わりに，3.5インチのディスクには位置決めホールがあり，それがインデックスホールと同じ役目を果たす。

実は3インチにも位置決めホールに相当する部分があり，別にインデックスホールがなくてもよかったのではないかという気がするが，しっかりとある。

次に「裏表」についてである。3.5，3インチでは不明だが，8，5インチではラベルを張る面がサイド1，反対側がサイド0である。1S，1Dなどの片面しか使わないドライブではサイド0だけを使っている（ただし例外もある）。

このサイドに関しては，3インチが変わっていて，1D用のドライブに対しては，（2D用の）ディスクを裏返して使うことができる。これがライトプロテクト用のスイッチが二つもある理由である。ただしX1の3インチドライブは2D用だから，そのようなことは関係なく，ディスクを裏返してセットすることはできない。

次に記録形式についてである。

まずは1S，2Dなどの形式について説明を行なっておく。最初に付いている数字は，

**「1」が片面**

**「2」が両面**

を意味している。

次にはアルファベットが1もしくは2文字付くが，これは，

**「S」＝単密度（Single density）**

**「D」＝倍密度（Double density）**

**「DD」＝倍トラック倍密度（Double track Double density）**

**「HD」＝高密度(High Density)**
の意味である。

まず，片面と両面の差であるが，これはそのものずばりであるから理解しやすいはずである。片面ドライブと両面ドライブの差は，要するにリード/ライト用のヘッドが一つだけか，それとも向かい合って二つ付いているかである。ただし両面ドライブの二つのヘッドはぴったりと向かい合っているわけではなく，少し前後に（数トラック分）ずれている。

次に「S」，「D」などについてである。これはまず，「S」と「その他」に分けられる。「S＝単密度」は別名「FM 記録方式」であり，「S」以外はすべて「MFM 記録方式」である。FM は「Frequency Modulation」（周波数変調方式）の略で，これはラジオの FM と同じである。つまり FM とはラジオだけのことではないのである。これに対して，MFM は「Modified Frequency Modulation 」である。Modified は，「修正された」というような意味である。詳しいことは他の参考書に譲るが，FM 記録方式ではクロックパルスというパルスが一定時間（5インチ1S，2Sなどの場合で8μs＝0.000008秒）ごとに書かれ，その間にデータ用のパルスがあれば「1」，なければ「0」である（データは1ビットだけ)。なぜこれが FM なのかといえば，全部「1」を書いた場合はパルスの間隔が 4μs になり，それに対して全部「0」を書いた場合はパルスの間隔は 8μs である。つまりパルスの来る周期が 4μs か 8μs かで1，0を区別しているのだ。だからやっぱり周波数変調＝FM である。

MFM はどうなのかというと，この方式は大胆にも，「前のデータが0で，なおかつ後ろのデータも0ならば，間にクロックパルスを入れる。それ以外の場合にはクロックパルスは書かない」ということになっている。このように修正してやると，「もっとも間隔の狭いパルスの間」は FM 方式のときの2倍になる。となれば，ディスクの磁性体とヘッドの性能の余裕は2倍になる（識別できる二つのパルスの間隔が「性能」である)。そこでほっと一息ついた磁性体とヘッドには気の毒だが，クロックパルスの間隔も入れた書き込み速度全体を2倍にしてやる。これで磁性体やヘッドの性能は基本的に同じままなのに（本当はそうは単純にいかないのだが)，倍のデータを読み書きできることになる。これが倍密度＝MFM である。早い話が，クロックビットを適当に間引くということなのである。「それじゃ，ディスクの回転と同期を取れなくなるじゃないか」という疑問が出てくるであろう（私もそう思う)。しかしディスクに書かれているのはデータだけではなく，あちこちに同期を取るための特殊なパルスも書かれていて，ちゃんと読み書きができるのである。これは後の「物理フォーマット」の所で説明する。

次に「DD」，「HD」についての説明である。「DD＝倍トラック倍密度」であるが，当然のことながら，ディスクにはトラックというものがあることは知っているだろう。これはバームクーヘンの年輪みたいなものである。このトラックに沿ってデータが書かれるのだが，倍トラックというのはこのトラックの間隔を半分にして，数を倍にしたものである。結局，めでたく容量は2倍になる。

最後が「HD」だが，これは8インチと同じ形式を5インチに持ってきたものである。トラック数は 2D と 2DD の中間であるが，パルス間隔が 2μs と，半分になり，1トラック当

たりのデータが多くなっている。それにより全体の容量は2DDの約1.5倍となる。

以上，あれこれ書いたが，結論として言えるのは，8インチ以外のディスクは**混沌に満ちている**ということである。おそらくこれは，急速に技術が向上しているものすべてに言えることだろうが，この先どーなるのか考えてみただけでも恐ろしい限りである。最近では2インチとかも出てきたし。

## トラック，セクタなどなど

まず用語について説明する。

**① サイド(side)**

表と裏の面のことである。サイド0，1がある。

**② トラック(track)**

1周分のデータを指す。

2Dは1サイド当たり40トラック，2DDは1サイド当たり80トラック，2HDは1サイド当たり77トラックである。なお，サイド0，1をまとめて数えて，「2Dは80トラック，2DDは160トラック」などと言う場合があるが，これは「誤解してください」と言っているのと同じである。できれば，「2Dは一つのサイド当たり40トラックで，サイドは二つある」と表現した方がよい(後述のシリンダを使う手もある)。なお，トラックの番号は外側から0番，1番……と数える。

ちなみに，サイド0と1では，実際の位置は各トラックで4本分(2DDでは8本)ずれている。サイド0の方が外側である。これはヘッドの位置がそもそも4本分（8本分）ずれているためである。このずれは，ディスクを使ううえではまったく意識する必要はない。

各ディスクにおけるトラック間距離は一定で，**表9-3**に示すとおりである。このことからも，FDDというものが高い精度を必要とするデリケートなものであることが分かるだろう。

**表9-3 各メディアにおけるトラック間距離**

| サ イ ズ | トラック間隔(mm) |
|---|---|
| 8インチ | 0.529 |
| 5インチ(2D)<br>(2DD，2HD) | 0.529<br>0.2646 |
| 3.5インチ(2D)<br>(2DD，2HD) | 0.375<br>0.1875 |
| 3インチ | 0.254 |

**③ シリンダ(cylinder)**

サイド0，1の同じ番号のトラックを併せて1シリンダと呼ぶ。2Dの場合は1枚のディスクに40シリンダあることになる。シリンダもトラックと同様に外側から0番，1番……と数える。

④ セクタ(sector)

1トラックを何個かに分解してセクタと呼ぶ。実際にデータを読み書きするのはこの部分である。5インチの2D，2DDでは普通1トラック（片面）当たり16セクタである。セクタはトラックと違い，1番，2番，……と番号が付けられている。なお，セクタの数は16個とは限らない。5インチの2Dのまともなフォーマットでは，1セクタの容量を256バイト，512バイト，1024バイトの中から選べる。それぞれの場合において，1トラック中のセクタの数は16，9，5となり，ディスク1枚当たりの容量は320Kバイト，360Kバイト，400Kバイトとなる。ただし裏技で，1トラックに17セクタを作ったりすることも可能である。

8インチ2Dと2HDの場合，セクタの容量と，1トラック（片面）当たりのセクタ数は，256バイト→26セクタ，512バイト→15セクタ，1024バイト→8セクタとなる。

当然ながら「容量は大きい方がよいのだから，1セクタは1024バイトにして，1枚当たりの容量を400Kバイトにすべきだ」と考えるかもしれないが，そうした場合，良いことばかりとは限らない。第一に読み書きは1セクタごとに行なわなければならないので，バッファは1Kバイト必要になる。さらには，1セクタを読み書きする時間も長くなる。結局ディスクの容量は2割増しになるという利点以外は，「1セクタ256バイトのフォーマットで，常に4セクタをまとめて読み書きしている」のと似たようなことになってしまうのである。さらには，他機種や，DOSとのデータ互換性なども問題になってくる。

# 本題に入る前に

本当はディスクエディタを載せるべきなのかもしれないが，敢えて載せないことにした。読み出して表示するだけであればBASICに付属の〝Device dump〞があるが，書き込みの際には別途にツールが必要になる。

# では，始める

まずはX1 HuBASICのソフト的なフォーマットについて書く。

一般的に，フロッピーディスクのフォーマットというと2とおりある。「物理フォーマット」と「ソフトフォーマット」である。物理フォーマットというのは，ディスケット上に作られた区画のようなものである。前述したように，1セクタの容量などが物理フォーマットによって決められる。BASICに付属している〝FORMAT & COPY. Uty〞が物理フォーマットをディスケットに書き込むわけである。物理フォーマットは1トラックごとに設定しなければならない点が，普通のデータ書き込みと異なる。普通，5インチの2D，2DDでは，1トラック分のフォーマットを設定するには，6.25 Kバイトのデータを必要とする。

ソフトフォーマットというのは，「ファイルの格納場所」や「IPL起動するファイルは，このような形式でディスケットに書かれていなければならない」などを定めるもので

ある。

X1のソフト的なフォーマットの要点は大体次の3点である。

① **ディレクトリの構造**

② **FAT(File Allocation Table)の構造**

③ **IPL 起動プログラムの指定**

①のディレクトリというのは，要するに「このディスクにはこんな名前のファイルが入っている」ということなどを記録してある部分である。ファイル名以外にも，ファイルの大きさや，書き込まれた日付などもある。つまり名簿のようなものである。②のFATは，現在ディスクのどの部分がどのように使われていて，どの部分が空いているかを示す部分である。③のIPLのプログラム指定は，ディレクトリの一種だと思えばよい。ではまず，ディレクトリの説明から始める。

ディレクトリの構造は**表 9-4**（turboのUSER'S MANUALより一部変更）である。一つのディレクトリは$32=20_H$バイトからなる。中身についての前に，レコード番号とクラスタ番号を説明しておく。レコード番号は5インチ2Dと3インチ2Dでは全部で$16\times2\times40=1280$個あるセクタに番号を付けたもので，

レコード番号＝(セクタ番号－1)
　　　　　　　＋(トラック番号×32)
　　　　　　　＋(サイド番号×16)

である。つまり，

トラック0，サイド0
トラック0，サイド1
トラック1，サイド0
トラック1，サイド1
⋮
⋮

と順番を付けて各セクタに番号を振ったものである。

クラスタというのは，16セクタをひとまとめにしたもので，FILES命令のときに出てくる「○△ Clusters free」のClusterである。第0クラスタは第0～15レコード，次の第1クラスタは第16～31レコード……となる。はっきり言って5インチ2Dと3インチ2Dでは，トラック1本分（片面）である。

そこでディレクトリの中身の説明であるが，第0バイトはファイルのアトリビュート(属性)を示している。各ビットごとに意味が割り振られているのだが，$00_H$と$FF_H$だけは特別で，$00_H$はKILLされたファイル，$FF_H$はここから先はまだ使用されたことのないディレクトリの領域であることを示す。つまり第0バイトが$FF_H$であるディレクトリ領域に出合ったら，後は見る必要がないということである。

各ビットの意味は表にあるとおりだが，turboで階層化ディレクトリが採用されたことにより，それまでX1で予備になっていた第7ビットが割り当てられた点に注意。また，第

表 9-4　ディレクトリの内容

| | 内　　　　　　　容 |
|---|---|
| 0 バイト目 | 種類を表す。　00 は KILL されたファイルまたは未使用領域。FF は使用ディレクトリテーブルの終わり。<br>bit 0 が 1 …… Bin ファイル(機械語で書かれたファイル)<br>bit 1 が 1 …… Bas ファイル(BASIC テキストで書かれたファイル)<br>bit 2 が 1 …… Asc ファイル(ASCII セーブされたファイル)<br>bit 4 が 1 …… FILES で表示しない：0 …表示する<br>bit 5 が 1 …… リードアフターライト ON ：0 …OFF<br>bit 6 が 1 …… 書き込み禁止ファイル：0 …書き込み OK<br>bit 7 が 1 …… 下位ディレクトリ<br>bit 3 は予備 |
| 1 バイト目～13 バイト目 | ファイル名(13 文字) |
| 14 バイト目～16 バイト目 | ユーザー指定 EXTENTION エリア(3 文字) |
| 17 バイト目 | パスワードのパック(無指定なら$20_H$の値) |
| 18 ・19 バイト目 | ファイルのバイト数(Bas および Obj のみ有効) |
| 20 ・21 バイト目 | ファイルのメインメモリ先頭アドレス(Obj のみ有効) |
| 22 ・23 バイト目 | ファイルのメインメモリ実行アドレス(Obj のみ有効) |
| 24 バイト目～28 バイト目 | 作成された年，月，曜日，時，分が書き込まれている。<br>例：'84 年 12 月 01 日土曜日，16 時 36 分<br>先頭から，　年年　月　曜　日日　時時　分分<br>　　　　　　84　C　6　01　16　36 |
| 29 バイト目～31 バイト目 | ファイル先頭　クラスタ値<br>29 バイト目　HIGH バイト(2D では 0)<br>30 バイト目　LOW バイト<br>31 バイト目　MIDDLE バイト(2D では 0) |

4 ビットは SET 命令で〝S〞，第 5 ビットは〝R〞，第 6 ビットは〝P〞の指定に対応する。

次に第 1 バイトからの 13 バイトであるが，これはファイル名である。ファイル名が 13 文字に満たない場合は，残りにスペースを詰めることになっている。ファイル名の中にはコントロールキャラクタを入れることも可能であるが，そうすると FILES での表示が狂ったり，単純には読み出せなくなったりする。昔，それをプロテクトに使ったソフトもあった。次に 3 バイトは拡張子と呼ばれるもので，「 . 」の後に付ける「Bas」とか「Uty」である。

第 17 バイトはパスワードである。パスワードとは，「SAVE　〝ファイル名；PASS〞」で指定できる。実際には文字列なのだが，計算によって 1 バイトに圧縮されてしまっている。だから，違うパスワードであっても，計算の結果同じ 1 バイトのデータになってしまうこともあり得る。パスワードを指定しなかった場合，この 1 バイトは $20_H$になっている。

第 18～23 バイトまでは特に説明する必要はないだろう。

第 24～28 バイトは日付である。サブ CPU から読み出したデータと同じ形式である。

第 29 バイトからの 3 バイトは，ファイル本体の先頭が書き込まれているクラスタ番号を指定している。3 バイトの数値で，High (上)，Low (下)，Middle (中) と，変則的な順序になっていることに注意。次の FAT の所で説明するが，ファイルの最小単位は 1 クラス

タである。つまり，1行しかないBAISCのプログラムをセーブしても1クラスタ＝4Kバイトを使ってしまう。

次にFATであるが，これは各クラスタの状態（使われているかどうか/つながりの順序など）を示している所である。X1のフォーマットでは2Dでは第14レコード，2DDでは第14，15レコード，2HDでは第28，29レコードにある。

まず。各クラスタの状態は次の三つに分かれる。

a） 使われていない

b） 使われていて，自分は最後ではない（別のクラスタに続く）

c） 使われていて，自分が最後のデータを持っている

第n番クラスタの状態は，FATの中の2バイトで表される。実際にどの2バイトかは，デバイスの容量によって少々複雑なのだが，128クラスタ以下（＝512Kバイト）の容量のデバイス（2Dなど）では単純で，「第nクラスタの状態は，FAT内の第nバイト目一つ」で分かるようになっている。そこで，ここではその場合に限定して説明する。

まずはその1バイトの値がa），b），c)のそれぞれにおいてどうなっているかというと，

a） → $00_H$

b） → $01_H$〜$7F_H$（次に続くデータを持つクラスタ番号）

c） → $80_H$〜$8F_H$（$7F_H$を引いた値が，そのクラスタ内で実際に使われているレコード数）

となっている。つまりチェーン状態を表しているのだ。2Dなどでは第0クラスタと第1クラスタ（レコード0〜32）はディレクトリと，FAT自身に使われているので，FATの先頭2バイトは常に，

$01_H$，$8F_H$，

となっている。すなわち「第0クラスタは1クラスタ全部が使われていて，第1クラスタに続く。第1クラスタは$8F_H-7F_H=16$レコードが実際に使われていて，このクラスタでチェーンは終わり」である。

最後のIPL起動プログラムの指定である。これは第0レコードに書き込まれて「ディレクトリ」に似ているものであるが，次の3点で普通のディレクトリと違う。

**① 先頭のアトリビュートは「Bin」の指定でなければならない**

**② ファイル名の拡張子は「Sys」でなければならない**

**③ 先頭クラスタ番号を指定する所には，代わりにレコード番号を書き込んでおく**

では最初にディレクトリの書き換えを行なってみる。INIT命令ですべてのファイルを消去したディスクに内容の違うBASICテキストを二つセーブする。そうするとBASICテキストのサイズによって若干異なるだろうが，ディレクトリの中身は大体**図9-1**のようになるはずである。そこで二つのディレクトリの「先頭クラスタ番号」（$1E_H$番目）と「ファイルの大きさ」（$12_H$，$13_H$番目）を交換して書き込んでやる。これで「ファイル名が交換された」のと同じことになる。

もう少しまともな用途としては，うっかり指定した後，忘れてしまったパスワードを解

図 9-1　ディレクトリの例

```
#001000=02 54 45 53 54 31 20 20 20 20 20 20 20 20 42 61 'ªTEST1        Ba
#001010=73 20 44 09 00 00 00 00 85 A5 05 05 01 00 02 00 's D‥‥‥‥※‥‥‥‥‥
#001020=02 54 45 53 54 32 20 20 20 20 20 20 20 20 42 61 'ªTEST2        Ba
#001030=73 20 64 02 00 00 00 00 85 A5 05 05 01 00 03 00 's d‥‥‥‥‥※‥‥‥‥‥
```

これらを交換すると「ファイル名が交換された」ことになる

除したりもできる。そのほか日付の書き換えやファイル名の変更もできる。

次に FAT もいじってみる。ここまでくるとやはり KILL されたファイルの復活が面白そうである。

まず，充分な大きさの BASIC テキストを作って欲しい。わざわざ打ち込まなくても RENUMBER と MERGE を繰り返せば長くすることができる。すなわち，

```
OPTION  SCREEN  2 : INIT "MEM :"
RENUM  1000 : SAVE "MEM : BT1", A
RENUM  2000 : SAVE "MEM : BT2", A
                  ⋮
                  ⋮
```

とした後で，各 ASCII ファイルを MERGE するなどしていけばよい。

充分に長い BASIC テキストを，INIT 命令ですべてのファイルを消去したディスクにセーブすると，ディレクトリと FAT は大体**図 9-2** のようになる。そこでそのファイルを KILL すると，おのおのは**図 9-3** のようになる。違っているのはディレクトリの第 0 バイトが $00_H$ で「消去されたファイル」を示していることと，対応する FAT がすべて $00_H$ に書き

図 9-2　ディレクトリと FATの例

```
#001000=02 52 45 56 49 56 41 4C 20 20 20 20 20 20 42 61 'ªREVIVAL      Ba
#001010=73 20 49 2D 00 00 00 00 85 A5 05 07 21 00 02 00 's I-‥‥‥‥※‥‥!‥‥


#000E00=01 8F 03 04 8D 00 00 00 00 00 00 00 00 00 00 00 '‥※‥※‥‥‥‥‥‥‥‥‥‥‥
#000E10=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥
#000E20=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥
```

図 9-3　ファイルを KILL した直後の ディレクトリと FAT の例

```
#001000=00 52 45 56 49 56 41 4C 20 20 20 20 20 20 42 61 '‥REVIVAL      Ba
#001010=73 20 49 2D 00 00 00 00 85 A5 05 07 21 00 02 00 's I-‥‥‥‥※‥‥!‥‥


#000E00=01 8F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '‥※‥‥‥‥‥‥‥‥‥‥‥‥‥
#000E10=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥
#000E20=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥
```

この部分が変わっている

換えられていることである。復活の第一歩に，まずディレクトリに残っているファイルサイズから，何クラスタ（何レコード）だったのかを計算する。例では1クラスタ＝4KB＝4096バイトより，2クラスタと14レコードだったということが分かる。よって，第2クラスタは第3クラスタへ，第3クラスタは第4クラスタへ，第4クラスタはそこで終わりで，14レコード分のデータを持っているから$7F_H+14=8D_H$となる。そのように書き換えていくと，結局図9-2のとおりになるはずである。それをディレクトリとFATの所に書き込んでやるとファイルは復活する。

しかし実際のファイルの復活はこのように**チョロイものではない**。何度もSAVE, KILLを繰り返したディスクだと，実際のクラスタのつながりが，2→9→10→12などのように飛び飛びになっている可能性があるのだ。「そのときはFATの$00_H$のクラスタを順に拾っていけばいいではないか」と思うかもしれないが，それは完全ではない。なぜなら，そのFATの$00_H$が復活したいファイルに対応しているとは限らないからである。3日前にKILLしたファイルに対応した$00_H$かもしれないのである。だから現実にファイルを復活するためには，「あーでもない，こーでもない」のパズルを繰り返さなければならない可能性が高いのである。なお，FATをいじりそこねると「Bad Record」というエラーが出る。場合によっては**暴走**することもあり得る。心するよーに。

ここでturboの階層化ディレクトリについて説明しておく。はっきり言って構造は簡単そのものである。つまり，MKDIRはもともとのディレクトリと同じサイズ＝1クラスタのファイルを作るだけなのである。CHDIRが実行されたなら，「そのディレクトリファイルを，ディレクトリのつもりで使う」だけなのである。すなわち，本質的には第1クラスタ（レコード番号16～32）に固定されていたディレクトリ領域を，任意のクラスタに変更できるようにしただけなのだ（本当はもっと複雑だが「本質」はそれだけのこと）。いろいろ試してみて，ダンプさせれば一目で分かるであろう。

次にIPL起動の指定を実際にやってみることとする。

BASIC（CZ-8FB01）を起動し，例によってINIT命令でファイルを全部消去したディスクを作る。そして，

```
POKE  &H012B, &H00, &H10
POKE  &H1053, &H00, &H10
```

を実行する。次に，

```
SAVEM  "ディスク番号：MONITOR.Sys", &H0000, &H149F, &H0000
```

でモニタとIOCS部分をディスクにセーブする。モニタを書き換えて縁起が悪いので，BASICを再起動する。次に何らかのツールを使って，ディレクトリの最初の1レコードを第0レコードへコピーする。後は第0レコードの30バイト目を，クラスタ番号($02_H$)からレコード番号（$20_H$）に書き換えればOK。なお，IPL起動のプログラムは連続したレコードに書かれている必要がある。

ここから先ではFDCなるLSIに詰め寄り，**抜く手も見せずに**フロッピーディスクの深遠へと向かうつもりである。

# とりあえずFDCについて

FDC (Floppy Disk Controller) とは，そのものずばりに，フロッピーディスクをコントロールするためのLSIである。FDCの種類は，「単密度専用」と「倍密度/単密度両用」の二つに分かれるが，「単密度専用」はすでに過去の遺物なので，「倍密度/単密度両用」についてのみ書くことにする。

倍密度/単密度両用FDCは，

**1)** FD1791ファミリ

**2)** μPD765ファミリ

**3)** その他

に分かれる。

FD1791はウエスタン・デジタル社が開発したFDCで，X1/X1 turboに使われているMB8877(富士通製)はFD1791ファミリに属する。それに対してμPD765は日本電気の開発したFDCである。当然のことながら，二つのファミリには互換性などというものは存在しない（それぞれが書き込んだディスケットはもちろんやりとりはできるが)。

ちなみに，FMシリーズのFDCはX1と同じMB8877, MZシリーズはMB8876を使っている。8877と8876の違いは，バスラインが正論理か，負論理かの違いだけである。具体的には，データの1/0が反転するのである。すなわち，X1でディスクに&HFF=&B11111111というデータを書き込み，そのディスクをMZのドライブに差し込むと&H00=&H00000000が読み出せるというわけである。またディスクの表裏 (SIDE) も逆である。試してみると分かるが，X1の第0レコード（シリンダ0，サイド0，セクタ1）はMZにとっては第16レコード（シリンダ0，サイド1，セクタ1）である。逆にMZの第0レコードはX1にとっては第16レコードである。

本題に入る。**表9-5**がMB8877/8876のコマンド，**表9-6〜9-8**が各フラグの意味，**表9**

**表9-5　MB8876/8877のコマンド**

| タイプ | | コマンド名称 | 動作 | 76543210 | X1での標準的な値 |
|---|---|---|---|---|---|
| I | 0 | リストア | トラック0へ，ヘッドを移動する | 0000hV$r_1 r_0$ | $02_H$=00000010 |
| | 1 | シーク | 所定のトラックへ，ヘッドを移動する | 0001hV$r_1 r_0$ | $1E_H$=00011110 |
| | 2 | ステップ | ヘッドを1トラック移動する | 001uhV$r_1 r_0$ | $3A_H$=00111010 |
| | 3 | ステップ・イン | ヘッドを1トラック内側へ移動する | 010uhV$r_1 r_0$ | $5A_H$=01011010 |
| | 4 | ステップ・アウト | ヘッドを1トラック外側へ移動する | 011uhV$r_1 r_0$ | $7A_H$=01111010 |
| II | 5 | リードデータ | ディスクのデータ(データフィールド)を読む | 100mSEC0 | $80_H$=10000000 |
| | 6 | ライトデータ | ディスク(データフィールド)へデータを書き込む | 101mSEC$a_0$ | $A0_H$=10100000 |
| III | 7 | リードアドレス | ディスクのIDフィールドを読む | 11000E00 | $C0_H$=11000000 |
| | 8 | リードトラック | ディスクの1トラック分の全データを読む | 11100E00 | $E0_H$=11100000 |
| | 9 | ライトトラック | ディスクへ1トラック分の全データを書き込む | 11110E00 | $F0_H$=11110000 |
| IV | 10 | フォースインタラプト | 割り込みを発生させる | 1101$I_3 I_2 I_1 I_0$ | $D0_H$=11010000 |

**-9, 9-10** がステータスレジスタの意味である。どーだ，これだけではちんぷんかんぷんだろう。そこで順に解説してゆくわけであるが，その前に**表 9-11** がある。これが X1/X1 turbo での FDC 関係の I/O アドレスである。アミかけ部分は X1 turbo 用である。これは後でまたねっとりとやる。

**表 9-6 TYPE I コマンド フラグ機能表**

| フラグ | 機能 |
|---|---|
| u | トラックレジスタ更新フラグ(u)は，ヘッド移動に際し，トラックレジスタを更新することを指示する。<br>u＝1：トラックレジスタを更新する。<br>u＝0：トラックレジスタを更新しない。 |
| h | ヘッドロードフラグ(h)は，コマンド実行開始時にヘッドをロードするか，ヘッドをメディアから離すかを指示する(X1では無意味)。<br>h＝1：コマンド実行開始時にヘッドロードする。<br>h＝0：コマンド実行開始時にヘッドを離す。 |
| V | トラック照合フラグ(V)は，ヘッド移動後，ディスクのトラック番号とトラックレジスタの照合を行なうかを指示する。<br>V＝1：トラックの照合を行なう。<br>V＝0：トラックの照合は行なわない。 |
| $r_1$, $r_0$ | ステップレートフラグ($r_1$, $r_0$)は，ステップパルス出力の間隔を指定する。 |

| | クロック | |
|---|---|---|
| $r_1$ $r_0$ | 1MHz (X1 ノーマル) | 2MHz (turbo の 2HD) |
| 0 0 | 6ms | 3ms |
| 0 1 | 12ms | 6ms |
| 1 0 | 20ms | 10ms |
| 1 1 | 30ms | 15ms |

**表 9-7 TYPE II, III コマンド フラグ機能表**

| フラグ | 機能 |
|---|---|
| m | マルチレコードフラグ(m)は連続セクタでリード/ライトするかを決める。<br>m＝1：連続セクタ(セクタ番号が増加する方向)でリード/ライトを行なう。<br>m＝0：単一セクタでリード/ライトを行なう。 |
| S | サイドフラグ(S)はサイド番号指定に使う。<br>S＝1：サイド番号の LSB が 1 のとき一致したものとみなす。<br>S＝0：サイド番号の LSB が 0 のとき一致したものとみなす。<br>(このフラグは，C フラグが 1 のときのみ有効) |
| E | ディレイフラグ(E)は HLT 信号のサンプリングのタイミングを指定する(X1では無意味)。<br>E＝1：HLD(ヘッドロード)信号を〝H〟とした後 15ms 待ち，HLT 信号をサンプリングする。<br>E＝0：HLD 信号を〝H〟とした後，直ちに HLT 信号をサンプリングする。 |
| C | サイド番号比較フラグ(C)はサイド番号と比較(チェック)するかどうかを指示する。<br>C＝1：サイド番号の比較を行なう。<br>C＝0：サイド番号の比較を行なわない。 |
| $a_0$ | アドレスマークフラグ($a_0$)はデータアドレスマークに何を書くか指定する。<br>$a_0$ ＝0：データアドレスマークに$(FB)_H$ (Data Mark)を書く。<br>$a_0$ ＝1：データアドレスマークに$(F8)_H$ (Deleted Data Mark)を書く。 |

**表 9-8 TYPE Ⅳ コマンド フラグ機能表**

| フラグ | 機能 |
|---|---|
| $I_0$ | $I_0$ = 1 で READY 入力の立ち上がりで IRQ 発生(IRQ = "H")。 |
| $I_1$ | $I_1$ = 1 で READY 入力の立ち下がりで IRQ 発生。 |
| $I_2$ | $I_2$ = 1 でインデックスパルス検出時に IRQ 発生。 |
| $I_3$ | $I_3$ = 1 で無条件で直ちに IRQ 発生 |

注意：X1/X1 turbo に使われている MB8877 の IRQ 信号は OPEN(どこにもつながっていない)だから，IRQ は何の機能も果たさない。

**表 9-9 ステータスレジスタの各ビットの意味**

| コマンド ＼ ステータスレジスタ | | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| TYPE I | すべてのコマンド | NOT READY | WRITE PROTECT | HEAD ENGAGED | SEEK ERROR | CRC ERROR | TRACK00 | INDEX | BUSY |
| TYPE II | リードデータ | NOT READY | 0 | RECODE TYPE | RECORD NOT FOUND | CRC ERROR | LOST DATA | DATA REQUEST | BUSY |
| | ライトデータ | NOT READY | WRITE PROTECT | WRITE FAULT | RECORD NOT FOUND | CRC ERROR | LOST DATA | DATA REQUEST | BUSY |
| TYPE III | リードアドレス | NOT READY | 0 | 0 | RECORD NOT FOUND | CRC ERROR | LOST DATA | DATA REQUEST | BUSY |
| | リードトラック | NOT READY | 0 | 0 | 0 | 0 | LOST DATA | DATA REQUEST | BUSY |
| | ライトトラック | NOT READY | WRITE PROTECT | WRITE FAULT | 0 | 0 | LOST DATA | DATA REQUEST | BUSY |
| TYPE Ⅳ | (他のコマンド実行中の場合) | (今まで実行していたコマンドのステータスビットと同様の意味) | | | | | | | 0 |
| | (実行中のコマンドのない場合) | NOT READY | WRITE PROTECT | HEAD ENGAGED | 0 | 0 | TRACK00 | INDEX | 0 |

**表 9-10 ステータスの意味**

| コマンド | ステータス | 意味 |
|---|---|---|
| TYPE I | NOT READY (bit 7) | NOT READY = 1 でディスクドライブが動作可能状態でないことを示す。 |
| | WRITE PROTECT (bit 6) | WRITE PROTECT = 1 でディスクへの書き込みが禁止されていることを示す。 |
| | HEAD ENGAGED (bit 5) | HEAD ENGAGED = 1 でヘッドがメディアに押しつけられていることを示す。 |
| | SEEK ERROR (bit 4) | SEEK ERROR = 1 でベリファイ動作が成功しなかったことを示す。これは，ID フィールドが検出されなかったか，ID フィールドのトラック番号とトラックレジスタの内容が一致しないことによる。 |
| | CRC-ERROR (bit 3) | CRC-ERROR= 1 で，ID フィールド読み出し時に読み出しエラーがあったことを示す。 |
| | TRACK00 (bit 2) | TRACK00 = 1 でディスクのヘッドがトラック 0 の上にあることを示す。 |
| | INDEX (bit 1) | INDEX = 1 でインデックスホールを検出したことを示す。 |

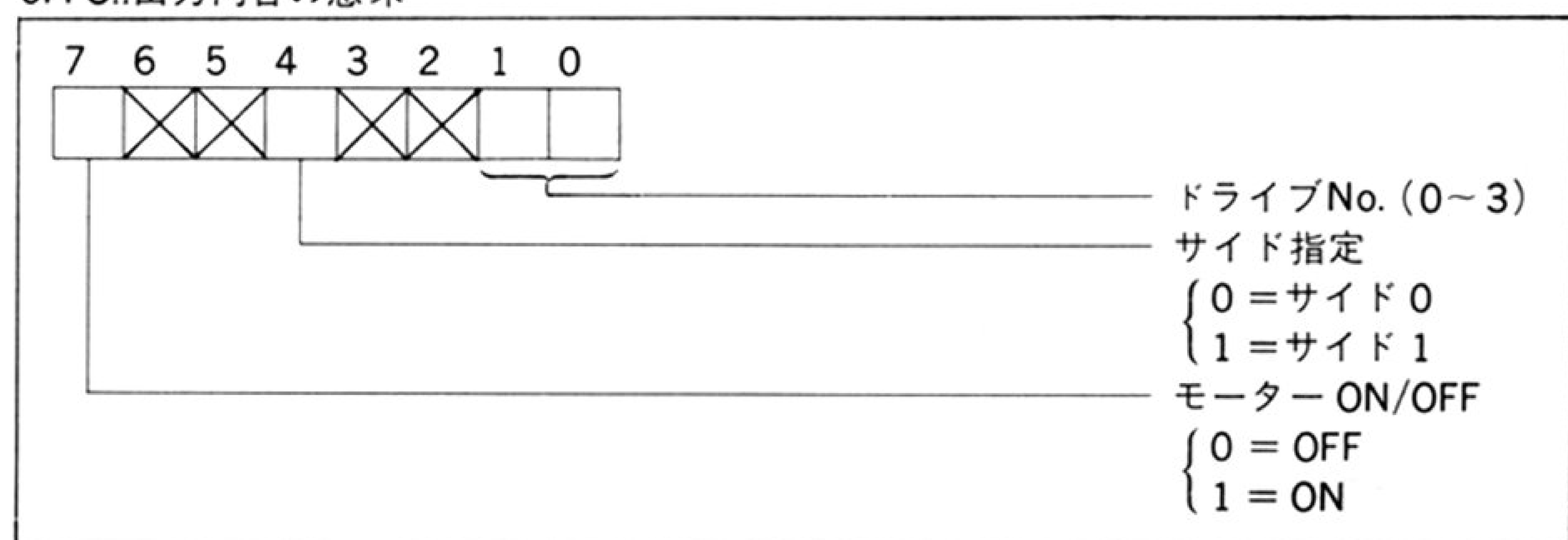
| | | |
|---|---|---|
| | BUSY (bit 0) | BUSY = 1 で FDC がコマンド動作中であることを示す。 |
| TYPE II/III | NOT READY (bit 7) | NOT READY = 1 でディスクドライブが動作可能状態でないことを示す。 |
| | WRITE PROTECT (bit 6) | WRITE PROTECT = 1 でディスクへの書き込みが禁止されていることを示す。 |
| | RECORD TYPE /WRITE FAULT (bit 5) | リード動作のとき，RECORD TYPE の表示として，デリーテッドデータマークのときセットされ，データマーク時リセットされる。ライト動作のとき，WRITE FAULT の表示として，書き込み動作が打ち切られたことを示す。 |
| | RECORD NOT FOUND (bit 4) | RECORD NOT FOUND = 1 で指定されたトラック番号，サイド番号，セクタ番号を持ち，正しく読み出された ID フィールドがなかったことを示す。 |
| | CRC ERROR (bit 3) | CRC ERROR = 1 でディスクから ID フィールドもしくはデータフィールドの読み出しにおいて，読み出しエラーがあったことを示す。 |
| | LOST DATA (bit 2) | LOST DATA = 1 で所要時間内に DR の読み出し，もしくは書き込みが行なわれなかったデータがあったことを示す。 |
| | DATA REQUEST (bit 1) | DATA REQUEST = 1 で，FDC がプロセッサに対して DR の読み出しもしくは書き込みを要求していることを示す。 |
| | BUSY (bit 0) | BUSY = 1 で FDC がコマンド実行中であることを示す。 |

表 9-11　FDC 関係の I/O アドレス

| | | |
|---|---|---|
| 0FF8H | ステータスレジスタ(STR)/コマンドレジスタ(CR) | IN/OUT |
| 0FF9H | トラックレジスタ(TR) | IN/OUT |
| 0FFAH | セクタレジスタ(SCR) | IN/OUT |
| 0FFBH | データレジスタ(DR) | IN/OUT |
| 0FFCH | FM 方式指定/ドライブ No.，サイド，モーター ON | IN/OUT |
| 0FFDH | MFM方式指定 | IN |
| 0FFEH | 1.6M タイプ(2HD)指定 | IN |
| 0FFFH | 500K(2D)・1M(2DD)切り換え | IN |

□の部分は turbo 用

0FFCH出力内容の意味




# TYPE Ⅰの実習

FDC, MB8877 を実際に使ってみるのである。まず，TYPE Ⅰのコマンドは機械語でなく，BASIC からでも使えるのでちょうどよい肩慣らしである。ただし，FDC を誤って動かすとディスクの内容がパーになったりすることがしばしばあるので，書き換わってもよいディスク以外はセットしておかないように。

FDC を使う前にしなければならないのが，ドライブのセレクトとモーター ON である。これには 0FFCH番地を使うのである。まずドライブ 1（"1："）に壊れてもよいフォ

ーマット済みのディスクを入れて，BASICから直接，

OUT　&HFFC，&H81

と打ち込んでいただきたい。ドライブ1のLEDがつき，ディスクが回り出したはずである。不幸にしてドライブ0しか持ってない人は，&H81の代わりに&H80である。ドライブ0が回り出すであろう。**冷たいようだが**以下同様。

いつまで回していても仕方がないので，止めることにする。

OUT　&HFFC，&H01（もしくは&H00）

と打ち込むと，約1秒後にディスクが止まる。なぜ1秒後かというと，これはディスクが回り出してから，読み書きできるようになるまでに時間がかかるからなのだ（わっかるかなー）。つまり，普通のディスクアクセス法として，一番自然な方法は，

**① ディスクを回し始める**

**② ディスクが規定の速度（毎分300回転）になるまで待つ**

**③ 読んだり，書いたりする**

**④ ディスクを止める**

となっている。

問題は，①〜④をやった後で，すぐにまた①〜④に飛んでくるケースが多いことにある。④で一旦ディスクが止まってしまっているから，（すぐさま）次にやってきたときに，またディスクがアクセス可能な状態になるまで待たなければならない。これでは時間の無駄が大きいので，④で「モーターOFF」の命令を受けても，すぐにはディスクを止めず，1秒間の遅延時間を置くのである。これによって，続けざまのディスクアクセスに時間の無駄が出ず，なおかつソフトウェアで面倒なことをしないで済むのである。めでたし，めでたし。

では，もう少し進んでみる。**リスト9-1**である。

**リスト9-1　TYPE Ⅰ 実習プログラム**

```
100 OUT &HFFC,&H81      :'MOTOR ON "1:"
110     GOSUB"WNBSY"
120 OUT &HFF8,&H2       :'RESTORE （第0トラックへシーク）
130     GOSUB"WNBSY"
140 OUT &HFFB,20        :'データレジスタ←目的トラック番号
150 OUT &HFF9,0         :'トラックレジスタ←現在のトラック番号
160 OUT &HFF8,&H1E      :'SEEK
170     GOSUB"WNBSY"
180 OUT &HFF8,&H5A      :'STEP IN
190     GOSUB"WNBSY"
200 OUT &HFFC,&H1
210 END
220 LABEL"WNBSY"
230 PRINT"]";SPACE$(3);
240 STAT=INP(&HFF8)      :'GET STATUS BYTE
250 PRINT RIGHT$("0"+HEX$(STAT),2);SPACE$(2);
260 IF STAT AND &H81 THEN 240
270 PRINT":":RETURN
```

まずは$FFC_H$←$81_H$で，ドライブ1のモーターをONする。GOSUB〝WNBSY〞で，ディスクが正しく定常状態になるまで待つ。次に$FF8_H$番地（コマンドレジスタ）へ$02_H$（表9-5のリストアコマンド）をOUTする。これで，ドライブ1のヘッドは第0トラック（一

番外側）へ移動してくれる。もちろん，ヘッドという重さのある「ブツ」が動くわけで多少の時間がかかるから，130 行の GOSUB "WNBSY" で動作の完了を待ってやる。

140～170 行は「シーク」である。このコマンドの使い方は，まず下準備として，$FFB_H$番地（データレジスタ）に目的トラック番号を OUT し，$FF9_H$番地（トラックレジスタ）へ現在のトラック位置を OUT する。次にコマンドレジスタへシークコマンド（$1E_H$）を送るのである。以上のことをすると，ドライブは「ククククッ」という音を立てて，目指すトラックへ動いてくれる。

その次はステッピインで，これは要するに 1 トラックだけ内側にシークする。

さて，220 行からのサブルーチンであるが，これは $FF8_H$（ステータスレジスタ）を見張りつつ，MSB と LSB（第 7 ビットと第 0 ビット）がともに 0 になるのを待つものである。第 7 ビットは「NOT　READY」，すなわち「まだ準備できてないよ」という意味。第 0 ビットは「BUSY」，すなわち「いま忙しいのっ！」もしくは「まだほかのことしてる最中でんがな。そないにせかさんといてや」という意味である。注意しておくが，I/O の $FF8_H$ 番地は，

OUT　したときはコマンドレジスタ
IN　　したときはステータスレジスタ

となっている。1 番地 2 役なのだ。

このプログラムでは，ステータスの内容をチェックする意味で，いちいちプリントさせている。実行結果は，**図 9-4** のようになるはずである。ちょっと見てみる。

**図 9-4 「リスト 9-1」の実行例**

```
]   A0  A0  A0  A0  A0  A0  A0  A0  A0  A0  A0  A0  A0  A0
A0  A0  A0  A0  A2  A0  A0  A0  A0  A0  A0  A0  A0  A0  A0
A0  A0  20  :
]   01  01  01  01  01  01  01  01  01  01  01  01  01  01
01  01  01  01  01  01  01  01  01  01  05  04  :
]   21  21  21  21  21  21  21  21  21  21  21  21  21  21
21  21  21  21  21  21  21  21  21  21  21  20  :
]   21  20  :
```

まず "]" は，「WNBSY が始まったよ」という意味である。最初のやつは，モーター ON から定常状態になるまで待っているわけだから，案の定 $A0_H = 10100000_B$ で第 7 ビットが 1 である。いっしょに第 5 ビットも立っているが，これは「ヘッドがディスクに押し付けられている」ということを示している――はずだが，どっこい，実は X1 ではこの信号は何の意味も持たないのであった。

ちょっと深入りするが，FDC ↔ FDD の間には HLD と HLT という信号がある。HLD は FDC から FDD への「ヘッドをディスクに押し付けなさい」という信号である。HLT は FDD から FDC への信号で「私こと FDD はディスクにヘッドを押し付けました。謹んでご報告いたします」というものである。しかし X1 では，HLD はオープン，HLT は 5V へプルアップされているのだ。すなわち，HLD 信号は何にも使われておらず，HLT は常に

Highのままである。これはどういうことかというと，5.25インチ以下のサイズのフロッピーディスクドライブでは一般的に「ヘッドは常にディスクに押し付けられているから」なのだ。X1のディスクでもそうなっている。よって第5ビットは無視すべきである。

さて，図9-4の2行目の中程に「A2」というのが1か所出てくる。これは第1ビットが立っているのである。表9-9から「INDEX」と分かる。すなわち，FDCはこの瞬間にインデックスホールを検出したのだ。後はずーっと$A0_H$で，最後に第7ビットが0になり「NOT READYではない」すなわち「用意できたよーん」となって“WNBSY”からリターンするのである。**言い遅れたが，**このようなステータスになるのは，直前（最後）にFDCに送られたコマンドがTYPE Iのコマンドだったからである。しかし，TYPEが何であっても「NOT READY」と「BUSY」は同じことだから，悩む必要はない。

次に“WNBSY”に来るのは，RESTORE（リストア：トラック0へのシーク）のときである。ステータスは$01_H=00000001_B$と，BUSYが並んでいる。最後の方は，$05_H$，$04_H$となっている。$05_H$は表9-9から分かるように「TRACK 00です　&　BUSY」である。ちょっと待て！　てめー，すでにTRACK 0に来てるんなら，RESTORE動作は終わってんじゃねーか。それなのにBUSYとは何事だ，と一瞬逆上するが，次の$04_H$でFDCはBUSYを取り下げている。むーん。それにしても第5ビットが0だ——一体どーなっているのだ。**ま，無視しておこう。**

次がシークコマンド後の“WNBSY”である。おっと，$21_H=00100001_B$だ。いやいや無視無視。えーと，こっちは平凡にBUSYが0になって終わりである。

最後がステップ・インである。これはシークと同じであるが，移動距離が短いので，あっという間に終わっている。

以上で図9-4の実感放送を終わる。

ではまず，シークコマンドについて少々補足しておく。シークコマンドでは，

　データレジスタ←目的トラック番号

　トラックレジスタ←現在のトラック番号

とセットするわけだが，どーしてこんな面倒なことをするのかというと，FDCが何台ものFDDを管理しているからなのだ。つまり，何台ものFDDを操作しているのだから，「現在のトラック番号」はX1の場合最大で四つある。本来なら，トラックレジスタは移動後の「トラック番号」を保持しているので，1台のドライブだけを動かしている場合は，目的トラック番号をデータレジスタにセットするだけでOKなのである。しかし現実はほとんどのシステムでFDDは2台以上あるから，DOS（もしくはディスクBASIC）は4バイトのワークエリアを持っていて，いちいちトラック番号を保持，書き込みしてやらなければならない。

トラックレジスタがトラック番号を持っていることを確認するためには，

```
175 PRINT  INP(&HFF9)
195 PRINT  INP(&HFF9)
```

を入れてやればよい。「20」，「21」と表示されるはずである。

次に大事な「フラグ」について説明する。リストアには，h，V，$r_1$，$r_0$の四つのフラグ

があり，各動作を指定している。まずは表 9-6 を見ると，

**h＝ヘッドロードフラグ**

とある。これはさっき言ったように無意味である。次に，

**V＝トラック照合フラグ**

である。これは実際にディスクを読んで，チェックするかどうかのフラグである。ディスクには「私は第△トラックの第×サイドの第○セクタです」という「ID」が書かれているのだ。これを読んで，ヘッドが本当に目的のトラックに達したかどうかのチェックを指定するフラグである。リスト 9-1 では $02_H$ だから V＝0 で，チェックは行なっていない。これについては，後で実験する。

次に $r_1$, $r_0$ であるが，これは二つで一組で，$00_B$〜$11_B$ までの四つの場合があり，これはステップレートを指定するものである。ステップレートは表 9-6 に示してある「時間」である。なぜこんなものがあるかというと，これは「FDD のヘッドを動かす速さ」に機種ごとに差があるからなのだ（一般に高価な FDD ほど速い)。FDC は FDD に「ヘッドを動かしなさい」という信号を送るのだが，その信号は「方向（内側へか外側へか)」と「動け」だけなのだ。FDD は一般に **タコだから，** FDC の方でタイミングを取ってやらないといけない。そのタイミングを指定するのが $r_1$, $r_0$ のステップレートなのである。たとえばリスト 9-1 のように $r_1$＝1，$r_0$＝0 を指定すると，FDC は FDD に対して 20 ms (0.02 秒) ごとに「動け」と指令を出すことになる。さて，実は X1 ではステップレートは別に 20 ms でなくてもよいのである。

3 インチ＝3ms
turbo の内蔵＝6ms
CZ-502F，503F＝6ms
古いタイプ（レバーのないやつ)＝20 ms

などとなっている。

そこで**リスト 9-2** が「ガリガリ」プログラムである。これは指定したステップレートで，ひたすらトラック 0 ↔ 39 間を動くというものである。しつこくやったり，性能以上の速度で動かそうとすると故障の原因になるから，ほどほどにやること。性能を超えた場合は「32」とか「30」(第 4 ビット＝シークエラーが 1）などが表示されるはずである。

後に残っているのは，STEP などの，

**u＝トラックレジスタ更新フラグ**

である。これは「ヘッドの移動にともなって，トラックレジスタの値を増減させるか，させないか」のフラグである。リスト 9-1 で 180 行の&H5A を&H4A にして，

```
195  PRINT  INP(&HFF9)
```

を追加すると「20」と表示されるはずである。実際には，ヘッドは第 21 トラック上にあるのだが，トラックレジスタはそうなっていないのである。

なぜこんなフラグがあるかというと，フロッピーディスクの始祖，IBM フォーマットには「欠陥トラック」というものがあるためなのだ。たとえば，ディスクをフォーマットしているときにどうしても正常に読み書きできないトラックを発見したとする。現在のパソ

リスト 9-2　ガリガリプログラム

```
100 INPUT "STEP RATE";S:IF (S<0) OR (S>3) THEN 100
110 OUT &HFFC,&H81        :'MOTOR ON "1:"
120     GOSUB"WNBSY"
130 OUT &HFF8,&H0+S       :'RESTORE （第0トラックへシーク）
140     GOSUB"WNBSY"
150 OUT &HFFB,39          :'データレジスタ←目的トラック番号
160 OUT &HFF9,0           :'トラックレジスタ←現在のトラック番号
170 OUT &HFF8,&H1C+S      :'SEEK
180     GOSUB"WNBSY"
190 IF INKEY$(0)="" THEN 130
200 OUT &HFFC,&H1
210 END
220 LABEL"WNBSY"
230 PRINT"]";SPACE$(3)
240 STAT=INP(&HFF8)       :'GET STATUS BYTE
250 PRINT RIGHT$("0"+HEX$(STAT),2);SPACE$(2);
260 IF STAT AND &H81 THEN 240
270 PRINT":":RETURN
```

コンではそんな場合には「エラーだよ，このディスクは使えません。ぴぴっ」とメッセージを出して，別のディスクを使うように要求するのだが，IBMフォーマットでは，「交替用トラック」が2トラック分（本当はシリンダと表現すべきだが）用意されていて，多少の欠陥のあるディスケットでも使えてしまうのである。ご丁寧にも，その欠陥のあるトラックを指定する方法も決まっている。この場合に，欠陥のあるトラックを飛ばしてフォーマットすると，実際のヘッドの位置と，ディスクに書かれている「ID」に違いが出てしまうことになる。そうなると，Vフラグでチェックしたときにシークエラーが出てしまうなどして都合が悪いので，uフラグを使って「つじつまを合わせる」のである。大昔は磁性体の品質が低かったり，ディスク（メディア）も高かったりしたから，そんなことをしたのだろうが，今はほとんど意味がないと言えるフラグである。しかし，IBMフォーマットがいかに**「金科玉条」**であるかのよい証拠でもある。

というところで，少々遊んでみることにする。まずはフォーマットされていない，まっさらのディスクを用意していただきたい。それをドライブ1に入れてもらう。その状態で，リスト9-1を実行すると，3番目のシークコマンドのステータスで，最後に$30_H$が出るはずである。すなわちシークエラーである。これは別にフォーマットされているディスクでも，ステップ・インで，u=0，V=1（つまりコマンドは&H4E）とすると同じことが起こる。トラックレジスタの値は「20」なのに，Vフラグに従ってチェックしてみると，第「21」トラックだった，というわけである。

TYPE Iコマンドで言い残したのは，STEPコマンドだけである。このコマンドを実行すると，ヘッドは内/外どっちに動くのかということであるが，「直前に動いた方」に動くということである。すなわち，FDCは「最後に動いた方向」を保持しているというわけである。

# TYPE Ⅱの実習

TYPE Ⅰでは機械語を使う必要がなかったが，TYPE Ⅱではさすがにそうはいかない。そこで，**リスト 9-3，9-4，9-5** である。例によってリスト 9-3 は機械語プログラムのアセンブルリスト，リスト 9-4 はダンプリスト，リスト 9-5 はそれを BASIC から使うためのプログラムである。

**BASIC は CZ-8FB01 の Ver 1.0**

でなければならない。なぜかというと，場合によっては $E000_H$ ～$F800_H$ ぐらいまでをワークエリアに使ってしまうので，turbo BASIC では動かない（暴走するかもしれない）からである。

リスト 9-3　ディスクアクセスプログラム

```
                              .Z80
                              .PHASE   0DF00H
                        ;
 0FF8                   CR    EQU      0FF8H    ;COMMAND REG.
 0FF8                   STR   EQU      0FF8H    ;STAT. REG.
 0FF9                   TR    EQU      0FF9H    ;TRACK REG.
 0FFA                   SCR   EQU      0FFAH    ;SECTOR REG.
 0FFB                   DR    EQU      0FFBH    ;DATA REG.
 0FFC                   MSDR  EQU      0FFCH    ;MOTOR,SIDE,DRIVE# SELECT
                        ;
 DF00   D5              START: PUSH    DE       ;SAVE DE FOR RETURN STATUS
 DF01   1A                    LD       A,(DE)   ;GET COMMAND No.
 DF02   13                    INC      DE
 DF03   D5                    PUSH     DE
 DF04   21 DF1A               LD       HL,JTA   ;HL=JUMP TABLE AREA
 DF07   16 00                 LD       D,0
 DF09   87                    ADD      A,A
 DF0A   5F                    LD       E,A      ;DE=A*2
 DF0B   19                    ADD      HL,DE
 DF0C   5E                    LD       E,(HL)
 DF0D   23                    INC      HL
 DF0E   56                    LD       D,(HL)   ;GET JUMP ADDRESS
 DF0F   EB                    EX       DE,HL    ;HL=JUMP ADDRESS
 DF10   D1                    POP      DE
 DF11   F3                    DI
 DF12   CD DF19               CALL     PATCH    ;CALL (HL)
                        ;
 DF15   FB                    EI
 DF16   D1                    POP      DE       ;GET STRING ADDRESS
 DF17   12                    LD       (DE),A   ;STORE LAST STATUS
 DF18   C9                    RET               ;RETURN TO BASIC
                        ;
 DF19   E9              PATCH: JP      (HL)     ;CALL (HL)
                        ;
                        ;     Jump Table Area
                        ;
                        ;TYPE I
 DF1A   DF32            JTA:  DW       RSTR     ;RESTORE
 DF1C   DF3C                  DW       SEEK     ;SEEK
 DF1E   DF55                  DW       STEP     ;STEP
 DF20   DF55                  DW       STPIN    ;STEP IN
 DF22   DF55                  DW       STPOT    ;STEP OUT
                        ;
                        ;TYPE II
 DF24   DF5F                  DW       READD    ;READ DATA
 DF26   DF84                  DW       WRITD    ;WRITE DATA
                        ;
                        ;TYPE III
 DF28   DF5F                  DW       READI    ;READ ID
 DF2A   DF5F                  DW       REDTR    ;READ TRACK
 DF2C   DF84                  DW       WRITR    ;WRITE TRACK
                        ;
                        ;TYPE IV
 DF2E   DFBC                  DW       FORI     ;FORCE INTERRUPT
                        ;
                        ;MOTOR,SIDE,DRIVE#
```

```
DF30    DFC6              DW      MSD      ;MOTER,SIDE,DRIVE#
                  ;
DF32    1A        RSTR:   LD      A,(DE)   ;COMMAND
DF33    01 0FF8           LD      BC,CR    ;BC=COMMAND REG.
DF36    ED 79             OUT     (C),A
DF38    CD DFD2           CALL    WNBSY    ;WAIT READY
DF3B    C9                RET
                  ;
DF3C    EB        SEEK:   EX      DE,HL
DF3D    7E                LD      A,(HL)   ;A=COMMAND
DF3E    23                INC     HL
DF3F    56                LD      D,(HL)   ;D=TRACK# TO SEEK
DF40    23                INC     HL
DF41    5E                LD      E,(HL)   ;E=CURRENT TRACK#
DF42    01 0FF9           LD      BC,TR
DF45    ED 59             OUT     (C),E    ;SET CURRENT TRACK#
DF47    01 0FFB           LD      BC,DR
DF4A    ED 51             OUT     (C),D    ;SET TRACK TO SEEK
DF4C    01 0FF8           LD      BC,CR
DF4F    ED 79             OUT     (C),A    ;SEND SEEK COMMAND
DF51    CD DFD2           CALL    WNBSY    ;WAIT 'NOT BUSY'
DF54    C9                RET
                  ;
DF55              STEP:
DF55              STPIN:
DF55    1A        STPOT:  LD      A,(DE)   ;GET COMMAND
DF56    01 0FF8           LD      BC,CR
DF59    ED 79             OUT     (C),A
DF5B    CD DFD2           CALL    WNBSY
DF5E    C9                RET
                  ;
DF5F              READD:
DF5F              READI:
DF5F    CD DFEA   REDTR:  CALL    SETSCT   ;A=COMMAND,SET SECTOR REG.
DF62    01 0FF8           LD      BC,CR    ;CR=STR
DF65    D9                EXX
DF66    01 0FFB           LD      BC,DR    ;BC'=DATA REG.
DF69    2A DFFE           LD      HL,(BUFAD)        ;HL'=BUFF ADD.
DF6C    D9                EXX
DF6D    ED 79             OUT     (C),A    ;SET COMMAND
DF6F    CD DFE4           CALL    WAIT1
DF72    ED 78     RED1:   IN      A,(C)    ;GET STATUS
DF74    0F                RRCA             ;CHECK BUSY
DF75    30 0B             JR      NC,RED2  ;END READ
DF77    0F                RRCA             ;CHECK DATA REQUEST
DF78    30 F8             JR      NC,RED1  ;NO REQUEST
                  ;
DF7A    D9                EXX
DF7B    ED 78             IN      A,(C)    ;GET DATA
DF7D    77                LD      (HL),A   ;STORE DATA
DF7E    23                INC     HL       ;INC BUFF. ADDR.
DF7F    D9                EXX
DF80    18 F0             JR      RED1     ;AGAIN
                  ;
DF82    07        RED2:   RLCA    ;BACK STAT (RRCA <-> RLCA)
DF83    C9                RET
                  ;
DF84              WRITD:
DF84    CD DFEA   WRITR:  CALL    SETSCT   ;A=COMMAND,SET SECTOR REG.
DF87    01 0FF8           LD      BC,CR    ;CR=STR
DF8A    D9                EXX
DF8B    01 0FFB           LD      BC,DR    ;BC'=DATA REG.
DF8E    2A DFFE           LD      HL,(BUFAD)        ;HL'=BUFF ADD.
DF91    D9                EXX
DF92    1E 00             LD      E,00H    ;COUNTER
DF94    16 02             LD      D,02H    ;MASK
DF96    ED 79             OUT     (C),A    ;SET COMMAND
DF98    ED 78     FLOOP:  IN      A,(C)    ;FIRST LOOP
DF9A    A2                AND     D
DF9B    C2 DFA4           JP      NZ,FIRST          ;FIRST DATA REQ.
DF9E    1D                DEC     E                 ;DEC COUNTER
DF9F    C2 DF98           JP      NZ,FLOOP          ;AGAIN
DFA2    18 06             JR      WRT1     ;QUIT FIRST LOOP
DFA4    D9        FIRST:  EXX
DFA5    7E                LD      A,(HL)   ;GET FIRST DATA
DFA6    ED 79             OUT     (C),A    ;WRITE 1 BYTE
DFA8    23                INC     HL       ;INC BUFF. ADDR.
DFA9    D9                EXX
DFAA    ED 78     WRT1:   IN      A,(C)    ;GET STATUS
DFAC    0F                RRCA             ;CHECK BUSY
DFAD    30 0B             JR      NC,WRT2  ;END WRITE
```

```
DFAF    0F                      RRCA             ;CHECK DATA REQUEST
DFB0    30 F8                   JR      NC,WRT1  ;NO REQUEST
                        ;
DFB2    D9                      EXX
DFB3    7E                      LD      A,(HL)   ;GET DATA
DFB4    ED 79                   OUT     (C),A    ;WRITE DATA
DFB6    23                      INC     HL       ;INC BUFF. ADDR.
DFB7    D9                      EXX
DFB8    18 F0                   JR      WRT1     ;AGAIN
                        ;
DFBA    07              WRT2:   RLCA    ;BACK STAT (RRCA <-> RLCA)
DFBB    C9                      RET
                        ;
DFBC    1A              FORI:   LD      A,(DE)   ;GET COMMAND
DFBD    01 0FF8                 LD      BC,CR
DFC0    ED 79                   OUT     (C),A
DFC2    CD DFD2                 CALL    WNBSY
DFC5    C9                      RET
                        ;
DFC6    1A              MSD:    LD      A,(DE)   ;GET DATA
DFC7    01 0FFC                 LD      BC,MSDR
DFCA    ED 79                   OUT     (C),A    ;SET IT
DFCC    B7                      OR      A        ;MOTOR ON?
DFCD    F0                      RET     P        ;MOTOR OFF THEN RET
DFCE    CD DFD2                 CALL    WNBSY    ;WAIT READY
DFD1    C9                      RET
                        ;
DFD2    C5              WNBSY:  PUSH    BC       ;SAVE BC
DFD3    06 20                   LD      B,20H
DFD5    10 FE           WNBSY0: DJNZ    WNBSY0   ;WAIT
DFD7    01 0FF8         WNBSY1: LD      BC,STR   ;DASAI
DFDA    ED 78                   IN      A,(C)
DFDC    4F                      LD      C,A      ;SAVE STATUS
DFDD    E6 81                   AND     81H      ;CHECK
DFDF    20 F6                   JR      NZ,WNBSY1
DFE1    79                      LD      A,C      ;GET STATUS
DFE2    C1                      POP     BC       ;GET BC
DFE3    C9                      RET
                        ;
DFE4    3E 07           WAIT1:  LD      A,7      ;WAIT ROUTINE
DFE6    3D              WAIT2:  DEC     A
DFE7    20 FD                   JR      NZ,WAIT2
DFE9    C9                      RET
                        ;
DFEA    EB              SETSCT: EX      DE,HL
DFEB    7E                      LD      A,(HL)   ;A=COMMAND
DFEC    23                      INC     HL
DFED    56                      LD      D,(HL)   ;D=SECTOR#
DFEE    01 0FFA                 LD      BC,SCR   ;BC=SECTOR REG.
DFF1    ED 51                   OUT     (C),D    ;SET SECTOR#
DFF3    F5                      PUSH    AF       ;SAVE COMMAND
DFF4    CD DFD2                 CALL    WNBSY
DFF7    F1                      POP     AF       ;GET COMMAND
DFF8    C9                      RET
                        ;
DFF9    00 00 00 00             DB      0,0,0,0,0        ;DUMMY
DFFD    00
                        ;
DFFE    E000            BUFAD:  DW      0E000H   ;BUFFER ADDRESS
                        ;
                                END
```

まずリスト 9-5の使い方からである。基本的なパターンは,

番号(0～11), コマンド, パラメータ…

をCHR$で文字列に変換した後にUSR0(～)である。詳しくは**表 9-12** に示す。11番だけはFDCへのコマンドではなく, FFC$_H$番地へのデータであることに注意。

なお, このプログラムは手抜きのお手本みたいなもので, ちゃんとしたエラー処理をしていない。たとえばディスクを抜いて, ドライブのモーターをONにしようとすると, 帰ってこないのである。そのときは心静かにリセットスイッチを押していただきたい。

では, **おもむろに**リスト 9-3の解説を始める。

リスト 9-4 「リスト 9-3」のダンプリスト(念のため:チェックには付録 A を参照のこと)

```
DF00 D5 1A 13 D5 21 1A DF 16 : 07
DF08 00 87 5F 19 5E 23 56 EB : C1
DF10 D1 F3 CD 19 DF FB D1 12 : 67
DF18 C9 E9 32 DF 3C DF 55 DF : 12
DF20 55 DF 55 DF 5F DF 84 DF : 09
DF28 5F DF 5F DF 84 DF BC DF : 7A
DF30 C6 DF 1A 01 F8 0F ED 79 : 2D
DF38 CD D2 DF C9 EB 7E 23 56 : 29
DF40 23 5E 01 F9 0F ED 59 01 : D1
DF48 FB 0F ED 51 01 F8 0F ED : 3D
DF50 79 CD D2 DF C9 1A 01 F8 : D3
DF58 0F ED 79 CD D2 DF C9 CD : 89
DF60 EA DF 01 F8 0F D9 01 FB : A6
DF68 0F 2A FE DF D9 ED 79 CD : 22
DF70 E4 DF ED 78 0F 30 0B 0F : 81
DF78 30 F8 D9 ED 78 77 23 D9 : D9
--------------------------------
SUM: 69 F3 1C A0 7A AD 85 E2 43E7
```

```
DF80 18 F0 07 C9 CD EA DF 01 : 6F
DF88 F8 0F D9 01 FB 0F 2A FE : 13
DF90 DF D9 1E 00 16 02 ED 79 : 54
DF98 ED 78 A2 C2 A4 DF 1D C2 : 2B
DFA0 98 DF 18 06 D9 7E ED 79 : 52
DFA8 23 D9 ED 78 0F 30 0B 0F : BA
DFB0 30 F8 D9 7E ED 79 23 D9 : E1
DFB8 18 F0 07 C9 1A 01 F8 0F : FA
DFC0 ED 79 CD D2 DF C9 1A 01 : C8
DFC8 FC 0F ED 79 B7 F0 CD D2 : B7
DFD0 DF C9 C5 06 20 10 FE 01 : A2
DFD8 F8 0F ED 78 4F E6 81 20 : 42
DFE0 F6 79 C1 C9 3E 07 3D 20 : 9B
DFE8 FD C9 EB 7E 23 56 01 FA : A3
DFF0 0F ED 51 F5 CD D2 DF F1 : B1
DFF8 C9 00 00 00 00 00 00 E0 : A9
--------------------------------
SUM: 6A 7F EE 56 A4 E0 A9 89 EC8E
```

リスト 9-5 ディスクアクセスサンプル

```
100 CLEAR &HDF00
110 MEM$(&HDF00,16)=HEXCHR$("D5 1A 13 D5 21 1A DF 16 00 87 5F 19 5E 23 56 EB")
120 MEM$(&HDF10,16)=HEXCHR$("D1 F3 CD 19 DF FB D1 12 C9 E9 32 DF 3C DF 55 DF")
130 MEM$(&HDF20,16)=HEXCHR$("55 DF 55 DF 5F DF 84 DF 5F DF 5F DF 84 DF BC DF")
140 MEM$(&HDF30,16)=HEXCHR$("C6 DF 1A 01 F8 0F ED 79 CD D2 DF C9 EB 7E 23 56")
150 MEM$(&HDF40,16)=HEXCHR$("23 5E 01 F9 0F ED 59 01 FB 0F ED 51 01 F8 0F ED")
160 MEM$(&HDF50,16)=HEXCHR$("79 CD D2 DF C9 1A 01 F8 0F ED 79 CD D2 DF C9 CD")
170 MEM$(&HDF60,16)=HEXCHR$("EA DF 01 F8 0F D9 01 FB 0F 2A FE DF D9 ED 79 CD")
180 MEM$(&HDF70,16)=HEXCHR$("E4 DF ED 78 0F 30 0B 0F 30 F8 D9 ED 78 77 23 D9")
190 MEM$(&HDF80,16)=HEXCHR$("18 F0 07 C9 CD EA DF 01 F8 0F D9 01 FB 0F 2A FE")
200 MEM$(&HDF90,16)=HEXCHR$("DF D9 1E 00 16 02 ED 79 ED 78 A2 C2 A4 DF 1D C2")
210 MEM$(&HDFA0,16)=HEXCHR$("98 DF 18 06 D9 7E ED 79 23 D9 ED 78 0F 30 0B 0F")
220 MEM$(&HDFB0,16)=HEXCHR$("30 F8 D9 7E ED 79 23 D9 18 F0 07 C9 1A 01 F8 0F")
230 MEM$(&HDFC0,16)=HEXCHR$("ED 79 CD D2 DF C9 1A 01 FC 0F ED 79 B7 F0 CD D2")
240 MEM$(&HDFD0,16)=HEXCHR$("DF C9 C5 06 20 10 FE 01 F8 0F ED 78 4F E6 81 20")
250 MEM$(&HDFE0,16)=HEXCHR$("F6 79 C1 C9 3E 07 3D 20 FD C9 EB 7E 23 56 01 FA")
260 MEM$(&HDFF0,16)=HEXCHR$("0F ED 51 F5 CD D2 DF F1 C9 00 00 00 00 00 00 E0")
270 DEFUSR0=&HDF00
280  D$=USR0(CHR$(11,&H81))       :'MOTOR ON
290  D$=USR0(CHR$(0,&H2))         :'RESTORE
300  D$=USR0(CHR$(1,&H1E,2,0))    :'SEEK
310 'D$=USR0(CHR$(2,&H3A))        :'STEP
320 'D$=USR0(CHR$(3,&H5A))        :'STEP IN
330 'D$=USR0(CHR$(4,&H7A))        :'STEP OUT
340  D$=USR0(CHR$(5,&H80,1))      :'READ DATA
350 'D$=USR0(CHR$(6,&HA0,1))      :'WRITE DATA
360 'D$=USR0(CHR$(7,&HC0,1))      :'READ ADD.
370 'D$=USR0(CHR$(8,&HE0,1))      :'READ TRACK
380 'D$=USR0(CHR$(9,&HF0,1))      :'WRITE TRACK
390 'D$=USR0(CHR$(10,&HD0))       :'FORCE INT.
400  D$=USR0(CHR$(11,&H1))        :'MOTOR OFF
410 END
```

DF00$_H$～DF19$_H$は，0～11番のうちの，どこへ飛び込むかを計算している。最初に DE レジスタを PUSH しているのは，後々ステータスを返すためである。DF11$_H$に「DI」があって割り込みを禁止しているが，これはディスクリード/ライトのようにタイミングが大事なプログラムでは必要なことである。DF12$_H$では DF19$_H$の「JP (HL)」とともに，「CALL (HL)」に相当することをやらせている。実際のルーチンは後回しにして，DF15$_H$～DF18$_H$を説明する。ディスクにアクセスして帰ってきたときに，A レジスタは最後に読み出したステータスレジスタの値を持っているようにしてある。その値を DE レジスタの指すアドレスにストアすると，このプログラムで呼び出した BASIC で，

表 9-12 「リスト 9-5」の使い方

| コ マ ン ド | 番号 | 第2パラメータ | 第3パラメータ | 第4パラメータ |
|---|---|---|---|---|
| リストア | 0 | コマンド | — | — |
| シーク | 1 | コマンド | 目的シリンダ No. | 現在のシリンダ No. |
| ステップ | 2 | コマンド | — | — |
| ステップ・イン | 3 | コマンド | — | — |
| ステップ・アウト | 4 | コマンド | — | — |
| リードデータ | 5 | コマンド | セクタ No. | — |
| ライトデータ | 6 | コマンド | セクタ No. | — |
| リードアドレス | 7 | コマンド | ダミー | — |
| リードトラック | 8 | コマンド | ダミー | — |
| ライトトラック | 9 | コマンド | ダミー | — |
| フォースインタラプト | 10 | コマンド | — | — |
| モーター，サイド，ドライブナンバー | 11 | データ | — | — |

D$＝USR0(～)

となっていると，D$の先頭の文字のASCIIコードとして受け取ることができる。便利，便利。

では，実行ルーチンの方であるが，たとえば$DF32_H$の RESTORE の実行ルーチンでは，「LD A，(DE)」によって，FDC に送るコマンドを A レジスタに拾い上げている。それをコマンドレジスタ(CR)に OUT して，WNBSY をコールする。よーするにこれは前述の BASIC で書いたプログラムと同じである。SEEK，STEP なども同様。

次に$DF5F_H$からの READD(リードデータ)，READI(リードアドレス)， REDTR(リードトラック) の3種混合ルーチンである。恐ろしいことにこれら三つのルーチンは完全に同じなのである。その結果， READI，REDTR では，必要がないのにセクタ番号を指定しているのである。実害はないから，大丈夫である。まず，「CALL SETSCT」で A レジスタにコマンドを拾い上げ，なおかつセクタレジスタ(SCR)にセクタ番号を OUT している。その先が少々複雑なのだが，BC レジスタにステータスレジスタ (SCR＝CR) のアドレスをセットし，EXX で裏レジスタにして，BC′にデータレジスタ (DR)，HL′にバッファアドレスをセットしている。ディスクのリード/ライトのためには，二つの違う I/O アドレスに高速にアクセスしなければならないのでこうしたのだが，STR と DR の上位アドレスはともに$0F_H$だから，どうしても裏レジスタを使わなければならないわけではない。結局は趣味の問題である。その後はコマンドを CR に OUT して，少々待つ。**なぜかよく知らないが，**とにかく待つ。それからはレジスタを表にしたり，裏にしたりしつつ，データを受け取るのである。$DF82_H$番地の RLCA は右にローテートしたステータスを左にローテートし直して，元に戻すものである。

次に$DF84_H$からの書き込む方であるが，$DF98_H$～$DFA9_H$間は，1バイトだけ書くルーチンである。なんでこんなことをしているかというと，ライトトラックのためなのだ。なぜかこうしないとライトトラックが成功しないのである。**理由はよく分からないが動け**

ばよいのだ。ほかは基本的にリードルーチンと同じである。

さて，リスト 9-5 を実行すると，ドライブ 1 に対してリストア動作した後 (290 行)，第 2 トラックへシークし (300 行)，第 1 セクタをリードし (340 行)，モーターを OFF する。データを読み込むアドレスは，$DFFE_H$番地からの 2 バイトに格納されているアドレスで，リスト 9-5 では $E000_H$になっている(260 行の右端)。D$の先頭バイトには，最後に読み込んだステータスレジスタの値が入っているから，

PRINT ASC (D$)

で，エラーチェックもできるようになっている。そこで早速リードデータの代わりに，370 行のリードトラックを実行してみていただきたい。「D$=～」の代わりに，「D1$=～」とでもしておくべきだろう。実行後，PRINT ASC (D1$) とすると，「4」が表示されるはずである。ステータス表 (表 9-9) を見ると，なんと「LOST DATA」である。これは結局どーゆーことかというと,「さっさと読み出さなかったから，データの取りこぼしがあるかもしれないよ」ということである。つまり，早い話がエラーである。

話せばわかる！ X1 では，どうしてもこうなるのである。これは仕方がないのである。私のプログラムのせいではないのだ。しかしここにも一筋の光明があるのだ。X1 turbo ならば,「LOST DATA」にはなるが，充分にデータは読み出せているのである。turbo 以外の機種では (全部を確認したわけではないが)，データがビットずれを起こしまくって，使いものにならない。

しかし，リードトラックよりも大事なことがあるのだ(と**逃げる**)。まずは TYPE IIコマンドのフラグについて説明しておく。

**m＝マルチレコードフラグ**

これは，複数のレコードを一度に読み書きするときに使うフラグである。後で実験する。

**S＝サイドフラグ**

これは C フラグといっしょに使うもので，読み出したディスクのサイドが 0 であるべきか，それとも 1 であるべきかを指定する。

**E＝ディレイフラグ**

これは，ヘッドがディスクに押し付けられたかどうかの信号のサンプリングタイミングを指定するもの。X1 では意味がない。

**C＝サイド番号比較フラグ**

これは V フラグがトラック (シリンダ) 番号を比較するかどうかのフラグであるのに対して，サイド番号を比較するかどうかのフラグである。しつこく言うが, S フラグといっしょに使う。

**$a_0$＝アドレスマークフラグ**

これはライトデータのときだけ指定できるものである。こいつに関してはサンプルを示した方が早いだろう。

では，TYPE IIの実習である。**図 9-5，9-6，9-7，9-8** である。

まずは m フラグをいじってみる。リスト 9-5 の 280 行からを図 9-5 のように書き換えて

いただきたい。まずは280行でモーターON，ドライブは〝1：″でサイドは0番である。リストアして，m＝1で第1セクタから読み始めている。325行で，それぞれのステータスレジスタを表示している。その値は，

リストア命令→4（TRACK 00＝1）

リードデータ命令→16(RECORD NOT FOUND)

となっている。モニタに飛んでダンプしてみると，しっかりと16セクタ分が読み込まれているはずである。そこで，BASICに帰り，

PRINT INP (&HFFA)

として，セクタレジスタの値を表示してみると「17」である。すなわちFDCはmフラグ＝1に従って，第1セクタから第16セクタまでを読み，挙げ句の果てに第17セクタまでを読もうとしたが，「ない袖は振れない」の法則に従って，当然のごとく失敗して，「RECORD NOT FOUND」を起こしたというわけである。このmフラグを使うと，連続セクタを読むという点ではなかなかに高速なのだが，惜しむらくは「いつ終わるのか指定できない」という欠点があるのだ。それでFDCは「第17セクタはどこだっ!?」と，少しの間捜し回るので，結局は時間をくってしまう。よって，正しく使うには，必要なだけを読んだかどうかをチェックしてやって，適当なところで，TYPE Ⅳのフォースインタラプトを使ってコマンドを打ち切ってやらなければならない。なかなかに世話のやけるmフラグであった。

図9-5　mフラグの例

```
m FLAG


280  D$=USR0(CHR$(11,&H81))       :'MOTOR ON
290  D0$=USR0(CHR$(0,&H2))        :'RESTORE
300  D5$=USR0(CHR$(5,&H90,1))     :'READ DATA
310 'D6$=USR0(CHR$(6,&HA0,1))     :'WRITE DATA
320  D$=USR0(CHR$(11,&H1))        :'MOTOR OFF
325 PRINT ASC(D0$),ASC(D5$),ASC(D6$)
330 END



    4        16       0
```

次にS，Cフラグである。図9-6は，平和な例で，280行でサイド0を指定しているから，S＝0（サイドは0ですか？），C＝1（チェックしてください）によって，ステータスは0という値になる。この逆が図9-7で，280行でサイド0を指定しているのに，S＝1（サイドは1ですか？），C＝1（チェックしてください）としているので，しっかり16＝RECORD NOT FOUNDが出ている。この場合はデータの読み込みは行なわれない。

図9-6　S, Cフラグの例 その1

```
S,C FLAG...S=0,C=1


280  D$=USR0(CHR$(11,&H81))       :'MOTOR ON
```

```
290  D0$=USR0(CHR$(0,&H2))          :'RESTORE
300  D5$=USR0(CHR$(5,&H82,1))       :'READ DATA
310 'D6$=USR0(CHR$(6,&HA0,1))       :'WRITE DATA
320  D$=USR0(CHR$(11,&H1))          :'MOTOR OFF
325 PRINT ASC(D0$),ASC(D5$),ASC(D6$)
330 END


        4         0         0
```

図 9-7　S, C フラグの例 その2

```
S,C FLAG...S=1,C=1


280  D$=USR0(CHR$(11,&H81))         :'MOTOR ON
290  D0$=USR0(CHR$(0,&H2))          :'RESTORE
300  D5$=USR0(CHR$(5,&H8A,1))       :'READ DATA
310 'D6$=USR0(CHR$(6,&HA0,1))       :'WRITE DATA
320  D$=USR0(CHR$(11,&H1))          :'MOTOR OFF
325 PRINT ASC(D0$),ASC(D5$),ASC(D6$)
330 END


        4         16        0
```

$a_0$フラグの実習が図 9-8 である。$a_0$フラグ＝1 で，ライトデータを実行すると，そのセクタにデリーテッドアドレスマークが書かれてしまうのだ。具体的にどこかというのは後程やるのである。これを検出するには，そのセクタを読むだけでよい。図 9-8 のように，ステータスレジスタが 32 となり，第 5 ビット（RECORD TYPE）が 1 になる。この場合にデータはちゃんと読まれるから，特にチェックしない限りデリーテッドアドレスマークが書かれていようがいまいが，同じことである。では，なぜこのようになっているのかというと，例によって「**IBM に聞いてくれ**」なのであった。

図 9-8　$a_0$ フラグの例

```
a0 FLAG


280  D$=USR0(CHR$(11,&H81))         :'MOTOR ON
290  D0$=USR0(CHR$(0,&H2))          :'RESTORE
300  D6$=USR0(CHR$(6,&HA1,1))       :'WRITE DATA
310  D5$=USR0(CHR$(5,&H80,1))       :'READ DATA
320  D$=USR0(CHR$(11,&H1))          :'MOTOR OFF
325 PRINT ASC(D0$),ASC(D5$),ASC(D6$)
330 END


        4         32        0
```

# TYPE Ⅲ

MB8877 の TYPE Ⅲコマンドにはリードアドレス，リードトラック，ライトトラックの

三つがあるわけである。そこでまずは，リードアドレスからである。

リードアドレスとは何かっつうと，ディスクに書かれている「ID フィールド」と呼ばれる部分を読み出すということである。正しくフォーマットされたディスクには，各セクタごとに ID フィールドとデータフィールドが1個ずつあるのだ。念のために言うと，「フィールド」とは「領域」とか「区画」という意味である。要するに平たく言えば，ID フィールドとはセクタの「ラベル」みたいなもので，データフィールドは「中身」である。

さて，ID フィールドの読み方であるが，**リスト 9-6** である。**縁起ものだから，**リスト 9-5 と同じ形式で載せておくことにする。RUN した後にモニタに飛んで，$E000_H$からをダンプすると最初の6バイトが大体**図 9-9** のようになっているはずである。この内訳は，

- 0バイト目＝シリンダ番号
- 1バイト目＝サイド番号
- 2バイト目＝セクタ番号
- 3バイト目＝セクタ長（0～3）
- 4バイト目＝CRC 上位
- 5バイト目＝CRC 下位

となっている。これらはすべて，フォーマット時に書き込まれたものである。セクタ番号は，最初に出会ったセクタのものである。よって，このままでは1～16番のうちどれが来るかは運まかせである。

セクタ長は，

0→1セクタ当たり 128 バイト

1→1セクタ当たり 256 バイト

2→1セクタ当たり 512 バイト

3→1セクタ当たり 1024 バイト

リスト 9-6　ID フィールドを読み出す場合の「リスト 9-5」の変更点100～270行は同じ

```
280  D$=USR0(CHR$(11,&H81))       :'MOTOR ON
290  D$=USR0(CHR$(0,&H2))         :'RESTORE
300 'D$=USR0(CHR$(1,&H1E,2,0))    :'SEEK
310 'D$=USR0(CHR$(2,&H3A))        :'STEP
320 'D$=USR0(CHR$(3,&H5A))        :'STEP IN
330 'D$=USR0(CHR$(4,&H7A))        :'STEP OUT
340 'D$=USR0(CHR$(5,&H80,1))      :'READ DATA
350 'D$=USR0(CHR$(6,&HA0,1))      :'WRITE DATA
360  D$=USR0(CHR$(7,&HC0,1))      :'READ ADD.
370 'D$=USR0(CHR$(8,&HE0,1))      :'READ TRACK
380 'D$=USR0(CHR$(9,&HF0,1))      :'WRITE TRACK
390 'D$=USR0(CHR$(10,&HD0))       :'FORCE INT.
400  D$=USR0(CHR$(11,&H1))        :'MOTOR OFF
410 END
```

図 9-9　ID フィールドの内容

```
:E000=00 00 02 01 AF 5F 00 00 /....ｧ_..
```

を指定している。普通は1セクタ256バイトだから「1」になっているはずである。最後の2バイトのCRCは，早い話がチェックサムみたいなものである。読み出すときのチェック用で，一定の方法で計算が行なわれる。FDCが勝手に計算してくれるので，別に心配する必要はない。もちろん，計算が合わなければ，「CRC ERROR」が発生するわけである。

というところで，もう一度図9-9を見ると，最初の2バイトは$00_H$，$00_H$で，つまりは第0シリンダの第0サイドなのである。そして，第2セクタに最初に出会った。そのセクタのタイプは1だから，ありふれた容量256バイトのフォーマットである。最後の2バイトのCRCの$AF_H$，$5F_H$は，CRCというわけである。

ここでわれわれその筋探検隊は，フロッピーディスクの深淵へと向かうのである。それはどーゆーことかというと，「ディスクの中で各セクタはどのように並んでいるのだろう」という疑問である。普通に考えるなら，1，2，3，……，16であるが，別にそうでなくてもよいはずである。そこで考えてみると，たしかTYPE Ⅰのコマンドを実行した直後ならINDEXホールを検出できるっつうことに思い当たるのであった。よって，インデックスホールを検出した後で，リードアドレスを立て続けに16回行なえば1周分の各セクタのIDフィールドが読めてしまうのである。

しかし，悲しいことにBASICでそれをやっても速度が追っつかないのである。結局はまたもや機械語を組まなければならない。うーむとつぶやきつつ**リスト9-7**と**リスト9-8**である。

**リスト9-7　IDフィールドの連続読み込み**

```
                                .Z80
                                .PHASE   0DE00H
                        ;
0FF8                    CR      EQU      0FF8H    ;COMMAND REG.
0FF8                    STR     EQU      0FF8H    ;STAT. REG.
0FFB                    DR      EQU      0FFBH    ;DATA REG.
                        ;
DE00    F3              START:  DI
                        ;
DE01    01 0FF8                 LD       BC,CR    ;CR=STR
DE04    ED 78           WINDEX: IN       A,(C)
DE06    E6 02                   AND      02H      ;CHECK INDEX
DE08    28 FA                   JR       Z,WINDEX
                        ;
DE0A    1A                      LD       A,(DE)
DE0B    57                      LD       D,A      ;D=COUNTER
DE0C    D9                      EXX
DE0D    01 0FFB                 LD       BC,DR    ;DATA REG.
DE10    2A DE37                 LD       HL,(BUFAD)       ;BUFF ADD.
DE13    D9                      EXX
                        ;
DE14    3E C0           AGAIN:  LD       A,0C0H   ;READ AD. COM.
DE16    ED 79                   OUT      (C),A    ;SEND COMMAND
DE18    3E 07                   LD       A,7
DE1A    3D              WAIT:   DEC      A
DE1B    20 FD                   JR       NZ,WAIT
DE1D    ED 78           RED1:   IN       A,(C)
DE1F    0F                      RRCA              ;CHECK BUSY
DE20    30 0B                   JR       NC,RED2  ;END READ
DE22    0F                      RRCA              ;CHECK DATA REQ.
DE23    30 F8                   JR       NC,RED1  ;NO REQ.
                        ;
DE25    D9                      EXX
DE26    ED 78                   IN       A,(C)
DE28    77                      LD       (HL),A   ;STORE DATA
DE29    23                      INC      HL
DE2A    D9                      EXX
```

```
DE2B    18 F0                   JR      RED1
                        ;
DE2D    07              RED2:   RLCA    ;BACK STAT
DE2E    D9                      EXX
DE2F    77                      LD      (HL),A   ;STORE STATUS
DE30    23                      INC     HL
DE31    D9                      EXX
DE32    15                      DEC     D        ;DEC COUNTER
DE33    20 DF                   JR      NZ,AGAIN
                        ;
DE35    FB                      EI
DE36    C9                      RET
                        ;
DE37    E000            BUFAD:  DW      0E000H
                        ;
                                END
```

**リスト 9-8 ID フィールドを連続読み込み表示する**

```
270 '100 - 270 ｷﾞｮｳ ﾊ ﾘｽﾄ9-5 ﾉﾓﾉｦ ﾂｶｳﾉﾃﾞｱﾙ
280 CLEAR &HDE00
290 MEM$(&HDE00,16)=HEXCHR$("F3 01 F8 0F ED 78 E6 02 28 FA 1A 57 D9 01 FB 0F")
300 MEM$(&HDE10,16)=HEXCHR$("2A 37 DE D9 3E C0 ED 79 3E 07 3D 20 FD ED 78 0F")
310 MEM$(&HDE20,16)=HEXCHR$("30 0B 0F 30 F8 D9 ED 78 77 23 D9 18 F0 07 D9 77")
320 MEM$(&HDE30,16)=HEXCHR$("23 D9 15 20 DF FB C9 00 E0 00 00 00 00 00 00 00")
330 DEFUSR1=&HDE00
340 COUNT=20
350 TR=0
360   D$=USR0(CHR$(11,&H81))     :'MOTOR ON
370   D$=USR0(CHR$(0,&H2))       :'RESTORE
380   D$=USR0(CHR$(1,&H1E,TR,0)):'SEEK
390   D$=USR1(CHR$(COUNT))       :'READ ADD.
400     FOR I=&HE000 TO &HE000+7*(COUNT-1) STEP 7
410       FOR J=I TO I+(7-1)
420         PRINT RIGHT$("0"+HEX$(PEEK(J)),2);" ";
430       NEXT:PRINT
440     NEXT
450   D$=USR0(CHR$(11,&H1))      :'MOTOR OFF
460 END
```

リスト 9-7 は，まず FDC のステータスレジスタを IN して，第 1 ビット(＝INDEX)が 1 になるまで待つ。その後，指定された回数だけバシバシとリードアドレスを実行して，最後の 1 バイトを読んだときのステータスといっしょにメモリに格納する。説明はこんなものでよいだろう。リスト 9-8 を見ていただきたい。これはリスト 9-7 の使い方である。100〜270 行には，リスト 9-6（9-5）の 100〜270 行を必要とする。

やっていることは単純で，まず 360 行でモーターを ON して，370 行でとりあえずリストアを実行してヘッドを第 0 トラックへ持ってくる。380 行ではお望みのトラックまでシークしている。トラック番号は TR に入っている。そこですかさず，

D$＝USR1(CHR$（読み出す回数))

である。最後のステータスもメモリに格納するから，1 回当たり 7 バイトのデータが取れるわけである。400 行からはそれを表示している。くれぐれも注意するが，USR1 の直前には TYPE Ⅰのコマンドを実行しておくこと。さもなくばリセットスイッチのお世話になってしまうのである。

ここで研究発表をするわけだが，それは**図 9-10** である。HuBASIC の場合は平和に 1〜16 が順に並んでいる。X1 CP/M は，第 0 と第 1 シリンダは HuBASIC と同じ，第 2 シリンダからは，図 9-10 に書いてあるようになっている。MZ-2000/2200 の CP/M では全シリンダにおいて図 9-10 のようになっている。

図 9-10　セクタの並び方

```
HuBASIC
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16
X1  CP/M (TRACK 2~39)
1  14  11  8  5  2  15  12  9  6  3  16  13  10  7  4
MZ  CP/M
1  3  5  7  9  11  13  15  2  4  6  8  10  12  14  16
```

そこで，どーしてこうなっているかである。

一般には第1セクタを読み書きした後は第2セクタを読み書きするわけであるが，CP/Mのようにハードウェアと密着していないソフトウェアなどは，その2回の読み書きの間に別の処理が入り，その間にディスクが少々回ってしまうのである。つまり，1〜16まで順に並んでいると，「第1セクタを読み終え，次に第2セクタを読もうとしても，すでに第2セクタはヘッドの位置を通り過ぎてしまっている」のだ。そうなると，ディスクがもう1回転するまで待たねばならない。前述のように，5インチや3インチの2D，2DDのディスクは1分間に300回転，すなわち1秒間に5回転しているから，1周分といえば0.2秒である。はっきり言って，これはとても長い時間なのだ。そこで，あらかじめ第1セクタと第2セクタの間を離しておくわけである。もちろんこれは第2セクタと第3セクタの間でも同じことである。第nセクタから何個目に第n＋1セクタがあるかを数えてみると(もちろん16→0と巡回するんだよ)，

HuBASIC→1個
X1 CP/M→5個
MZ CP/M→8個

となっている。この1，5，8，の数字はスキューファクタと呼ばれる。これと同じような意味を持つものにインターリーブファクタというものもあって，これは隣り合ったセクタ番号の差である。

HuBASIC→2－1＝1
X1 CP/M→14－1＝13
MZ CP/M→3－1＝2

となる。MZ CP/Mの場合は15→2，16→1の所で乱れている点に注意。ま，どーでもいいけどね。

この先は少々その筋の話になるのだが，**その筋のディスクにその筋すると，**その筋のセクタ番号があったりする。たとえば，第0セクタとか，第17セクタである。さらには第1セクタが26個あったりなどの**凶器攻撃**もある。ここらへんのことは自由研究としておいて，私は次にリードトラックとライトトラックを説明するのである。

X1/X1 turboでリードトラックを実行すると，どちらでもLOST DATAエラーが起きるが，turboではデータが使いものになる理由を説明する。これはディスクのインターフェイスの性能に関係するのである。だからturboにCZ-500FやCZ-800Fを接続してリードトラックを実行しても，使いものになるデータが読めるのである（所々でビットずれは起

こすが，それはいかなる場合でも起こることである)。これは，どうやらVFOというクロックビットとデータビットを分離する回路の性能に関係するらしい。turboはその回路の性能がいま一なので，取り出したデータは使えるがLOST DATAエラーが発生し，turbo以外は，いま三なのでエラーが起きるだけではなく，データも狂いまくるわけである。しかし，リードトラック以外では問題が起きないので，あんまり強く文句を言うわけにはいかない―――が，私は文句を言うぞっ！

**「文句，文句，文句，文句，文句」**

では，ライトトラックを説明する。そうすればリードトラックも自動的に理解できる。ライトトラックの機能は，言うまでもなく，ディスクに物理フォーマットを書き込むことにある。物理フォーマットは，1トラック(片面分)当たり約6.25Kバイト(6400バイト)のデータを，ライトトラックコマンドを使ってディスクに書き込むことにより指定される。その6.25Kバイトの内訳は，「はいっ，ここからここまではIDフィールドですよ。そいで，ここからの256バイト分はデータです。第1セクタの次には第2セクタが続きます」などということの指定である。では早速であるが，「ISOタイプ5.25インチフォーマット」に使う6.25Kバイト分のデータを**表9-13**に示す。この表は，参考文献10に大いにお世話になっている。

さて，実はこいつはなかなかの喰わせものなのである。よって説明する。

まず，表9-13から分かるようにフォーマットは三つの部分に分かれている。GAP1と各セクタのフォーマット（複数個ある）とGAP4である。つまり，1トラックのフォーマットはGAP1で始まり，最後にGAP4で終わるということである。しかし主役は，その間に挟まれた「セクタのフォーマット」である。1トラック（片面）に，256バイトの容量のセクタが16個ある普通のフォーマットでは，

GAP1，各セクタ×16，GAP4

となっているわけである。

では，その普通のフォーマットについて，インデックスホールの位置から順に説明する。

● GAP1

インデックスホールの位置から，$4E_H$の相当するデータが32バイト分である。このほかにもGAP2，3，4があるが，いずれもフロッピーディスクドライブのばらつき（回転速度のズレなど）に対処するためのものである。

●最初のセクタ（第1セクタとは限らない）

ここから表9-13の下の部分へ行く。

セクタは次の部分からなる。

**Sync**

$00_H$が12個書かれている。これは同期を取るためのもの。

**AM1**

ID ADDRESS MARKである。これは3個の$A1_H$(実はちょっと違う)と1個の$FE_H$からなる。$A1_H$は，普通のデータではなく，ミッシングクロック(Missing Clock）を含んで

表 9-13　ISO タイプ 5.25 インチフォーマット

| 1 セクタ当たりの容量 | GAP1 | 各セクタのフォーマット | GAP4 |
|---|---|---|---|
| 256 バイト | $4E_H \times 32$ | ①× 16 | $4E_H \times 266$ |
| 512 バイト | | ②× 9 | $4E_H \times 296$ |
| 1024 バイト | | ③× 5 | $4E_H \times 208$ |
| 実際のデータ(MB8877 用) | 同　上 | 下を見よ | 同　上 |

(第 t シリンダ，第 s サイド，第 n セクタ用のフォーマット：実際はこれが複数個並んでいる)

| | 容　量 | Sync | AM1 | | ID | | | | CRC | GAP 2 | Sync | AM 2 | | DATA | CRC | GAP 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ① | 256 | $00_H \times 12$ | $(A1_H) \times 3$ | $FE_H \times 1$ | t | s | n | 1 | (?)×2 | $4E_H \times 22$ | $00_H \times 12$ | $(A1_H) \times 3$ | $FB_H$ または $F8_H$ | 256 バイト | (?)×2 | $4E_H \times 54$ |
| ② | 512 | | | | | | | 2 | | | | | | 512 バイト | | $4E_H \times 84$ |
| ③ | 1024 | | | | | | | 3 | | | | | | 1024 バイト | | $4E_H \times 116$ |
| 実際のデータ (MB8877 用) | | 同　上 | $F5_H \times 3$ | 同　上 | 同　上 | | | | $F7_H \times 1$ | 同　上 | 同　上 | $F5_H \times 3$ | 同　上 | 同　上 | $F7_H \times 1$ | 同　上 |

ID ADDRESS MARK

DATA MARK もしくは DELETED DATA MARK

ID フィールド

データフィールド

1 セクタ分のフォーマット

いる。これは英語的な言い方で，実にその筋なので言い直すと，よーするにクロックビットが 1 個足りないのだ。$A1_H$は 2 進数だと，

$10100001_B$

となり，これは MFM 記録方式の規則から，0 と 0 の間にクロックビットが入るので，普通ならば，

```
1 0 1 0 0 0 0 1
       ↑ ↑ ↑
```

の 3 か所にクロックビットが入るはずである。ところがどっこい，この $A1_H$の場合は，真ん中のクロックビットがないのである。だから表 9-13 の下の部分では ($A1_H$) のように，カッコでくくってある。これは後で出てくる AM2 でも同じである。($A1_H$)の後の $FE_H$は普通の $FE_H$である。さて，下の方に「実際のデータ」という欄があるが，ここに「$F5_H$」とか書いてある。つまり MB8877 にこのクロックビットの抜けている ($A1_H$) を書かせるときは，$A1_H$の代わりに $F5_H$を送ってやればよいのだ。MB8877 はライトトラックを実行中に，$F5_H$，$F6_H$，$F8_H$などが送られてくると特別扱いするのである。この点については後でまとめてやる。

**ID**

ここには，シリンダ番号，サイド番号，セクタ番号，セクタのタイプ (大きさ) などが書かれている。リスト 9-6 の「リードアドレス」で読み出したデータのうちの最初の 4 バイト分である。

**CRC**

このCRCは，AM1の最後の$FE_H$とIDの計5バイトに対するCRCである。MB8877では$F7_H$を1バイト送っただけで，2バイトのCRCを勝手に計算して書いてくれる。

**GAP2**

GAP1と同じようなもの。

**Sync**

さっき説明した。

**AM2**

ここにある3個の($A1_H$)はAM1と同じである。その次に$FB_H$が来るのだが，これは先程やったライトデータの$a_0$フラグと関係しているのだ。普通は$FB_H$ (DATA MARK) なのだが，$a_0=1$でライトデータを行なうと$F8_H$ (DELETED DATA MARK) になってしまうのである。

**DATA**

普通読み書きするのはこの部分である。

**CRC**

AM2の$FB_H$（もしくは$F8_H$）とデータを併せた部分のCRCである。

**GAP3**

GAP1，2と同じ。

これで1セクタ分が終わりである。

● 2～16番目のセクタ

これは「最初のセクタ」と基本的に同じ形式。ただセクタ番号だけが違う。

● GAP4

$4E_H$が，インデックスホールまで続いているのである。

以上がトラック（しつこいようだが片面分）のフォーマットである。途中で，「$F5_H$を送ると（$A1_H$）が書かれる」などと書いたが，そのようなことになる値の一覧表を**表9-14**に示す。表9-14Ⓐの中では（$C2_H$）とかが出てくるが，これは「IBMタイプフォーマット」の中で使われるものなのである。実は5.25インチのフロッピーディスクのフォーマットには，メジャーなものが2とおりあるのだ。二つのフォーマットの違いは大きくはなく，**表9-15**を表9-13の代わりに使うと「IBMタイプ」になる。見て分かるとおりに，これはGAP4を短くしてGAP1を増やし，さらにGAP0とかINDEX MARKとかを入れてしまったものである。ただし全体のバイト数は変わっていないことに注意していただきたい。二つのフォーマットの違いは，参考文献10によると，ISOタイプの方がエラー発生の可能性が減るということらしい。たしかに表を見比べるとIBMタイプはGAPを短くしてINDEX MARKなどというものを入れているのだから，FDCがどちらのフォーマットも読めるのであれば，ゆとりのあるISOタイプの方が有利であろう。ちなみにHuBASICとS-BASIC (MZ-2000) はISOタイプ，MZ-2000のCP/Mは大胆にもIBMタイプのようである。

表 9-14　ライトトラック(フォーマット)時のデータの意味

Ⓐ MFM 動作時

| MB8877/76に送るデータ | ディスクに書かれるデータ | 意味 |
|---|---|---|
| $00_H$ ～ $F4_H$ | $00_H$ ～ $F4_H$ | 普通のデータ |
| $F5_H$ | ($A1_H$) | AM1，AM2 の前提(CRC ジェネレータをプリセットする) |
| $F6_H$ | ($C2_H$) | INDEX MARK の前提(CRCジェネレータをプリセットする) |
| $F7_H$ | CRC | 内部で計算された CRC (2 バイト)を書く |
| $F8_H$ | $F8_H$ | AM2 の中では DELETED DATA MARK |
| $F9_H$<br>$FA_H$ | $F9_H$<br>$FA_H$ | 普通のデータ |
| $FB_H$ | $FB_H$ | AM2 の中では DATA MARK |
| $FC_H$ | $FC_H$ | ($C2_H$)×3 の後で INDEX MARK |
| $FD_H$ | $FD_H$ | 普通のデータ |
| $FE_H$ | $FE_H$ | AM1 の中では ID ADDRESS MARK |
| $FF_H$ | $FF_H$ | 普通のデータ |

Ⓑ FM 動作時

| MB8877/76に送るデータ | ディスクに書かれるデータ | | 意味 |
|---|---|---|---|
| | データパターン | クロックパターン | |
| $00_H$ ～ $F4_H$ | $00_H$ ～ $F4_H$ | $FF_H$ | 普通のデータ |
| $F5_H$ | 禁止 | | |
| $F6_H$ | | | |
| $F7_H$ | CRC | $FF_H$ | 内部で計算された CRC (2 バイト)を書く |
| $F8_H$ | $F8_H$ | $C7_H$ | AM2 の中では DELETED DATA MARK(CRCジェネレータをプリセットする) |
| $F9_H$<br>$FA_H$ | $F9_H$<br>$FA_H$ | $C7_H$ | CRC ジェネレータをプリセットする |
| $FB_H$ | $FB_H$ | $C7_H$ | DATA MARK (CRC ジェネレータをプリセットする) |
| $FC_H$ | $FC_H$ | $D7_H$ | INDEX MARK (CRC ジェネレータをプリセットする) |
| $FD_H$ | $FD_H$ | $FF_H$ | 普通のデータ |
| $FE_H$ | $FE_H$ | $C7_H$ | ID ADDRESS MARK (CRC ジェネレータをプリセットする) |
| $FF_H$ | $FF_H$ | $FF_H$ | 普通のデータ |

注) FM 方式の場合，普通のデータのクロックパターンは「$FF_H$」である。

**表 9-15　IBM タイプ 5.25 インチフォーマット**

| 1セクタ当たりの容量 | GAP0 | Sync | INDEX MARK | | GAP1 | 各セクタのフォーマット | GAP4 |
|---|---|---|---|---|---|---|---|
| 256 バイト | | | | | | ①×16 | $4E_H$×152 |
| 512 バイト | $4E_H$×80 | $00_H$×12 | ($C2_H$)×3 | $FC_H$×1 | $4E_H$×50 | ②×9 | $4E_H$×132 |
| 1024 バイト | | | | | | ③×5 | $4E_H$×94 |
| 実際のデータ | 同　上 | 同　上 | $F6_H$×3 | 同　上 | 同　上 | 表9-13の下の部分を見よ | 同　上 |

そこで，どどーんとディスクをフォーマットするプログラムが**リスト 9-9** である（注釈を見ると turbo BASIC だということが分かるが，実際は turbo BASIC では動かないので注意)。1010〜1170 行は例の機械語部分（リスト 9-5 の 110〜270 行）をリナンバーしたものである。二，三説明すると，まずこのプログラムは CZ-8FB01 でなければ動かない。それから 1340〜1380 行は，USR0 を実行してエラーが起きた場合に 5 回まで試してみるためのものである。このプログラム（機械語部分）だと，どうもタイミングがいまいちらしく，変数 STAT に返ってくるステータスがたまに「2」になってしまう。これは「DATA REQUEST だけ」ということなのだ。「BUSY」にはなっていないのである。困ったものだ。しかし，たまにだけだから許すことにしたのである。次に 1860〜1900 行であるが，これはシリンダ番号，サイド番号，セクタ番号，セクタ容量，のメモリへの書き込みを行なっている。データに対応するアドレスはサブルーチン "MF-SUB" によって配列 VAR（〜）に入れられてあるのだ。RUN すると，ドライブ "1：" に入っているディスクを，のんびりとフォーマットする。各トラック（片面）ごとにステータスを表示し，0 でなかったら数度（5 回まで）リトライするようになっている。

**リスト 9-9　ディスクフォーマットプログラム(turbo BASIC では動かない)**

```
1000 CLEAR &HC000
1010 MEM$(&HDF00,16)=HEXCHR$("D5 1A 13 D5 21 1A DF 16 00 87 5F 19 5E 23 56 EB")
1020 MEM$(&HDF10,16)=HEXCHR$("D1 F3 CD 19 DF FB D1 12 C9 E9 32 DF 3C DF 55 DF")
1030 MEM$(&HDF20,16)=HEXCHR$("55 DF 55 DF 5F DF 84 DF 5F DF 5F DF 84 DF BC DF")
1040 MEM$(&HDF30,16)=HEXCHR$("C6 DF 1A 01 F8 0F ED 79 CD D2 DF C9 EB 7E 23 56")
1050 MEM$(&HDF40,16)=HEXCHR$("23 5E 01 F9 0F ED 59 01 FB 0F ED 51 01 F8 0F ED")
1060 MEM$(&HDF50,16)=HEXCHR$("79 CD D2 DF C9 1A 01 F8 0F ED 79 CD D2 DF C9 CD")
1070 MEM$(&HDF60,16)=HEXCHR$("EA DF 01 F8 0F D9 01 FB 0F 2A FE DF D9 ED 79 CD")
1080 MEM$(&HDF70,16)=HEXCHR$("E4 DF ED 78 0F 30 0B 0F 30 F8 D9 ED 78 77 23 D9")
1090 MEM$(&HDF80,16)=HEXCHR$("18 F0 07 C9 CD EA DF 01 F8 0F D9 01 FB 0F 2A FE")
1100 MEM$(&HDF90,16)=HEXCHR$("DF D9 1E 00 16 02 ED 79 ED 78 A2 C2 A4 DF 1D C2")
1110 MEM$(&HDFA0,16)=HEXCHR$("98 DF 18 06 D9 7E ED 79 23 D9 ED 78 0F 30 0B 0F")
1120 MEM$(&HDFB0,16)=HEXCHR$("30 F8 D9 7E ED 79 23 D9 18 F0 07 C9 1A 01 F8 0F")
1130 MEM$(&HDFC0,16)=HEXCHR$("ED 79 CD D2 DF C9 1A 01 FC 0F ED 79 B7 F0 CD D2")
1140 MEM$(&HDFD0,16)=HEXCHR$("DF C9 C5 06 20 10 FE 01 F8 0F ED 78 4F E6 81 20")
1150 MEM$(&HDFE0,16)=HEXCHR$("F6 79 C1 C9 3E 07 3D 20 FD C9 EB 7E 23 56 01 FA")
1160 MEM$(&HDFF0,16)=HEXCHR$("0F ED 51 F5 CD D2 DF F1 C9 00 00 00 00 00 00 E0")
1170 DEFUSR0=&HDF00
1180 DEFINT A-Z:DIM VAR(40),SQ(40)
1190 DN=1                   :'ドライブ番号
1200 AD=&HE000              :'フォーマット用データのアドレス
1210 PRINT"MAKING DATA":GOSUB"MAKE-FORMAT":'フォーマット用データを作る
1220 '
1230 DUMMY=INP(&HFFF)                        :'2D,2DDモードにする
1240 OUT &HFFC,&H80+DN:GOSUB"WNBSY"          :'モーターON
1250 OUT &HFF8,&H0 :GOSUB"WNBSY"             :'リストア
1260 OLDTR=0
1270 RESTORE"TYPE":READ SECTORTYPE
1280 '
1290 FOR CYLINDER=0 TO 39
1300   GOSUB"SEEK":OLDTR=CYLINDER:'SEEK
1310   FOR SIDE=0 TO 1
1320     OUT &HFFC,&H80 OR DN OR SIDE*&H10         :'裏表の設定
```

```
1330      GOSUB"SET-FORMAT"
1340      TRY=0
1350      IF TRY>5 THEN BEEP:PRINT:PRINT"ERROR":GOTO 1430
1360      GOSUB"WTR"                              :'WRITE TRACK
1370      PRINT STAT;":";                         :'PRINT STATUS
1380      IF STAT<>0 THEN TRY=TRY+1:GOTO1350       :'RETRY
1390    NEXT
1400 NEXT
1410 'DMA$=HEXCHR$("83"):GOSUB"SETDMA"
1420 PAUSE 1
1430 OUT &HFFC,&H3 AND DN:'モーターＯＦＦ
1440 DUMMY=INP(&HFFF)                       :'２Ｄ，２ＤＤモードにする
1450 END
1460 '
1470 LABEL"SEEK"
1480 OUT &HFFB,CYLINDER  :'データレジスタ
1490 OUT &HFF9,OLDTR     :'トラックレジスタ
1500 OUT &HFF8,&H10      :'シークコマンド
1510 GOSUB"WNBSY"
1520 RETURN
1530 '
1540 LABEL"WNBSY"
1550 CT=0
1560 IF CT>1000 THEN PRINT"DISK?":GOTO 1410
1570 IF INP(&HFF8) AND &H81 THEN CT=CT+1:GOTO 1560
1580 RETURN
1590 '
1600 LABEL"SETDMA"
1610 FORI=1TOLEN(DMA$):OUT &H1F80,ASC(MID$(DMA$,I,1)):NEXT
1620 RETURN
1630 '
1640 LABEL"WTR"
1650 D2$=USR0(CHR$(9,&HF0,1)):STAT=ASC(D2$):RETURN
1660 DMA$=HEXCHR$("83 79 00 C0 AF 28 14 28 8D FB 0F 92 CF 05 CF 87")
1670 GOSUB"SETDMA"
1680 OUT &HFF8,&HF0      :'ライトトラックコマンド
1690 S=INP(&HFF8):IF S AND 1 THEN 1690
1700 STAT=S:RETURN
1710 '
1720 LABEL"MAKE-FORMAT"
1730 VAR(0)=0:'CLEAR COUNTERS
1740 RESTORE"GAP1":GOSUB"MF-SUB"      :'MAKE GAP1
1750 RESTORE"TOTAL SECTOR":READ TSEC :'HOW MANY SECTORS ?
1760 FOR SECT=1 TO TSEC
1770   RESTORE"SECTOR":GOSUB"MF-SUB" :'MAKE SECTOR FORMAT
1780 NEXT
1790 RESTORE"GAP4":GOSUB"MF-SUB"      :'MAKE GAP4
1800 RESTORE"SQUE"
1810 FOR SECT=1 TO TSEC
1820   READ SQ(SECT)
1830 NEXT
1840 RETURN
1850 '
1860 LABEL"SET-FORMAT":'SET ID FIELD
1870 FOR P=1 TO VAR(0)
1880   MEM$(VAR(P),4)=CHR$(CYLINDER,SIDE,SQ(P),SECTORTYPE)
1890 NEXT
1900 RETURN
1910 '
1920 LABEL"MF-SUB"
1930 READ C:IF C=0 THEN RETURN:' n Byte,DATA ﾉ ｹｲｼｷ
1940 READ D$:IF D$<>"!" THEN D=VAL(D$):GOTO 1970
1950 '!..MEANS "VARIABLE DATA"
1960 VAR(0)=VAR(0)+1:VAR(VAR(0))=AD:D=99:'INC COUNTER, STORE ADDRESS
1970 IF C<256 THEN MEM$(AD,C)=STRING$(C,D):AD=AD+C:GOTO1930
1980 MEM$(AD,255)=STRING$(255,D):AD=AD+255:C=C-255:GOTO1970
1990 '
2000 LABEL"GAP1"
2010 DATA 32,&H4E,0
2020 LABEL"SECTOR"
2030 DATA 12,0,3,&HF5,1,&HFE,4,!
2040 DATA 1,&HF7,22,&H4E,12,0,3,&HF5,1,&HFB,256,&HE5,1,&HF7,54,&H4E,0
2050 LABEL"GAP4"        :'                         ^                ^
2060 DATA 266,&H4E,0   :'<-!!!                    !!!              !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 16            :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
2110 LABEL"TYPE"
2120 DATA 1             :'<-!!! 1 SECTOR 256 Byte つぅことでんがな
```

さて，このプログラムは実にその筋なので，2000 行から後を**リスト 9-10** と替えると，なんと 1 セクタ 512 バイトのフォーマットとなるのである。さらには**リスト 9-11** に替えると，1 セクタが 1024 バイトになってしまうのだ。ディスク 1 枚で 400K バイトだぞ。さらに追い撃ちだっ。3 インチドライブや X1 turbo などのドライブ (CZ-800F/801F 以外) だとヘッドは第 42 シリンダまで動くから，1290 行の「39」を「42」にするとディスク 1 枚で 430 K バイトだだだだだっ！　持ってけドロボーなのである。これは「オーバートラック」と呼ばれる **ヒネリ技である。** もちろん第 40 シリンダ以降の書き込み，および書き込まれたデータには，シャープ株式会社は何の保証もしない「その筋トラック」であるから覚悟して使うように。

**リスト 9-10　ディスクフォーマットプログラム　1 セクタ＝512 バイト用変更点**

```
2000 LABEL"GAP1"
2010 DATA 32,&H4E,0
2020 LABEL"SECTOR"
2030 DATA 12,0,3,&HF5,1,&HFE,4,!
2040 DATA 1,&HF7,22,&H4E,12,0,3,&HF5,1,&HFB,512,&HE5,1,&HF7,84,&H4E,0
2050 LABEL"GAP4"        :'                       ^                 ^
2060 DATA 296,&H4E,0  :'<-!!!                   !!!               !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 9             :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5,6,7,8,9
2110 LABEL"TYPE"
2120 DATA 2             :'<-!!! 1 SECTOR 512 Byte つぅことでんがな
```

**リスト 9-11　ディスクフォーマットプログラム　1 セクタ＝1024 バイト用変更点**

```
2000 LABEL"GAP1"
2010 DATA 32,&H4E,0
2020 LABEL"SECTOR"
2030 DATA 12,0,3,&HF5,1,&HFE,4,!
2040 DATA 1,&HF7,22,&H4E,12,0,3,&HF5,1,&HFB,1024,&HE5,1,&HF7,116,&H4E,0
2050 LABEL"GAP4"        :'                        ^                  ^
2060 DATA 208,&H4E,0  :'<-!!!                    !!!                !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 5             :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5
2110 LABEL"TYPE"
2120 DATA 3             :'<-!!! 1 SECTOR 1024 Byte つぅことでんがな
```

なお，このプログラムには仕掛けがしてあるので，turbo の場合は少々変更すると DMA を使ったディスクフォーマットルーチンになる。それについてはまた後程。

次に**リスト 9-12，9-13** である。これは G-RAM ↔ディスク間の直接読み書きルーチンなのだ。リスト 9-12 では，書き込みルーチンがなかなか味わい深い。この部分では，DATA REQUEST 信号に素早く応答できるように A′ (裏レジスタ) に，次に書き込むべきデータを用意してあるのだ。タイミングが難しいケースにおける，いわゆる一つの解決例になっている。

リスト 9-12　G-RAM↔ディスク直接 R/W プログラム

```
                                 .Z80
                                 .PHASE   0DD00H
                           ;
  0FF8                     CR    EQU      0FF8H    ;COMMAND REG.
  0FF8                     STR   EQU      0FF8H    ;STAT. REG.
  0FF9                     TR    EQU      0FF9H    ;TRACK REG.
  0FFA                     SCR   EQU      0FFAH    ;SECTOR REG.
  0FFB                     DR    EQU      0FFBH    ;DATA REG.
  0FFC                     MSDR  EQU      0FFCH    ;MOTOR,SIDE,DRIVE# SELEC
T
                           ;
  DD00   D5                START: PUSH    DE
  DD01   DD E1                   POP      IX              ;IX=DE
  DD03   DD 66 01                LD       H,(IX+1)        ;H=COMMAND
  DD06   DD 6E 02                LD       L,(IX+2)        ;L=SECTOR#
  DD09   16 FB                   LD       D,LOW(DR)
  DD0B   1E F8                   LD       E,LOW(STR)
  DD0D   D9                      EXX
  DD0E   DD 4E 04                LD       C,(IX+4)
  DD11   DD 46 05                LD       B,(IX+5)        ;BUFF ADD.
  DD14   ED 78                   IN       A,(C)    ;GET FIRST 1 Byte
  DD16   08                      EX       AF,AF'   ;A'=FIRST DATA
  DD17   D9                      EXX
  DD18   DD 7E 00                LD       A,(IX+0)        ;GET 'R' or 'W'
  DD1B   F3                      DI
                           ;
  DD1C   FE 52                   CP       'R'
  DD1E   28 0A                   JR       Z,READD
                           ;
  DD20   FE 57                   CP       'W'
  DD22   28 2C                   JR       Z,WRITD
                           ;
  DD24   DD 36 00 FF             LD       (IX+0),0FFH     ;SET ERROR
  DD28   FB                      EI
  DD29   C9                      RET
                           ;
  DD2A                     READD:
  DD2A   CD DD77           RAGAIN: CALL   SETFDC   ;SET SCT#,COM.
                           ;
  DD2D   ED 78             RED1:  IN      A,(C)    ;GET STATUS
  DD2F   0F                      RRCA              ;CHECK BUSY
  DD30   30 0E                   JR       NC,RED2  ;END READ
  DD32   0F                      RRCA              ;CHECK DATA REQ.
  DD33   30 F8                   JR       NC,RED1  ;NO REQ.
                           ;
  DD35   4A                      LD       C,D      ;BC=DR
  DD36   ED 78                   IN       A,(C)    ;READ DATA
  DD38   D9                      EXX
  DD39   ED 79                   OUT      (C),A    ;STORE DATA
  DD3B   03                      INC      BC       ;NEXT ADD.
  DD3C   D9                      EXX
  DD3D   4B                      LD       C,E      ;BC=STR
  DD3E   18 ED                   JR       RED1
                           ;
  DD40   07                RED2:  RLCA              ;BACK STAT
  DD41   DD 77 00                LD       (IX+0),A        ;STORE STAT
  DD44   E6 9F                   AND      9FH      ;CHECK ERROR
  DD46   20 06                   JR       NZ,RED3  ;QUIT READ
  DD48   2C                      INC      L        ;IX+2=SCTNO
  DD49   DD 35 03                DEC      (IX+3)   ;IX+3=COUNT
  DD4C   20 DC                   JR       NZ,RAGAIN
  DD4E   FB                RED3:  EI
  DD4F   C9                      RET
                           ;
  DD50                     WRITD:
  DD50   CD DD77           WAGAIN: CALL   SETFDC   ;SET SCT#,COM.
                           ;
  DD53   ED 78             WRT1:  IN      A,(C)
  DD55   0F                      RRCA              ;CHECK BUSY
  DD56   30 10                   JR       NC,WRT2  ;END WRITE
  DD58   0F                      RRCA              ;CHECK DATA REQ.
  DD59   30 F8                   JR       NC,WRT1  ;NO REQ.
                           ;
  DD5B   4A                      LD       C,D      ;BC=DR
  DD5C   08                      EX       AF,AF'   ;A'=DATA
  DD5D   ED 79                   OUT      (C),A    ;WRITE DATA
  DD5F   D9                      EXX
  DD60   03                      INC      BC       ;INC ADD.
  DD61   ED 78                   IN       A,(C)    ;GET NEXT DATA
```

```
DD63      08                          EX      AF,AF'   ;A'=NEXT DATA
DD64      D9                          EXX
DD65      4B                          LD      C,E      ;BC=DR
DD66      18 EB                       JR      WRT1
                             ;
DD68      07                 WRT2:    RLCA             ;BACK STAT
DD69      DD 77 00                    LD      (IX+0),A          ;STORE STAT
DD6C      B7                          OR      A        ;CHECK ERROR
DD6D      20 06                       JR      NZ,WRT3  ;ERROR THEN END
DD6F      2C                          INC     L        ;IX+2=SCTNO
DD70      DD 35 03                    DEC     (IX+3)   ;IX+3=COUNT
DD73      20 DB                       JR      NZ,WAGAIN
DD75      FB                 WRT3:    EI
DD76      C9                          RET
                             ;
DD77      7D                 SETFDC:  LD      A,L      ;IX+2=SCTNO
DD78      01 0FFA                     LD      BC,SCR
DD7B      ED 79                       OUT     (C),A
DD7D      01 0FF8                     LD      BC,CR    ;CR=STR
DD80      ED 61                       OUT     (C),H
DD82      3E 07                       LD      A,7
DD84      3D                 WAIT:    DEC     A
DD85      20 FD                       JR      NZ,WAIT
DD87      C9                          RET
                             ;
                                      END
```

リスト 9-13　G-RAM↔ディスク直接転送プログラム

```
270 '100 - 270 ｷﾞｮｳ ﾊ ﾘｽﾄ9-5 ﾉﾓﾉｦ ﾂｶｳﾉﾃﾞｱﾙ
280 CLEAR &HDD00
290 MEM$(&HDD00,16)=HEXCHR$("D5 DD E1 DD 66 01 DD 6E 02 16 FB 1E F8 D9 DD 4E")
300 MEM$(&HDD10,16)=HEXCHR$("04 DD 46 05 ED 78 08 D9 DD 7E 00 F3 FE 52 28 0A")
310 MEM$(&HDD20,16)=HEXCHR$("FE 57 28 2C DD 36 00 FF FB C9 CD 77 DD ED 78 0F")
320 MEM$(&HDD30,16)=HEXCHR$("30 0E 0F 30 F8 4A ED 78 D9 ED 79 03 D9 4B 18 ED")
330 MEM$(&HDD40,16)=HEXCHR$("07 DD 77 00 E6 9F 20 06 2C DD 35 03 20 DC FB C9")
340 MEM$(&HDD50,16)=HEXCHR$("CD 77 DD ED 78 0F 30 10 0F 30 F8 4A 08 ED 79 D9")
350 MEM$(&HDD60,16)=HEXCHR$("03 ED 78 08 D9 4B 18 EB 07 DD 77 00 B7 20 06 2C")
360 MEM$(&HDD70,16)=HEXCHR$("DD 35 03 20 DB FB C9 7D 01 FA 0F ED 79 01 F8 0F")
370 MEM$(&HDD80,16)=HEXCHR$("ED 61 3E 07 3D 20 FD C9 00 00 00 00 00 00 00 00")
380 DEFUSR2=&HDD00
390 '
400 GOSUB"GSMPL"
410   D$=USR0(CHR$(11,&H81))      :'MOTOR ON
420   D$=USR0(CHR$(0,&H2))        :'RESTORE
430   TR0=0:FTR=0:FSCT=1:COM$="W"+CHR$(&HA0):HI=&H40
440   SCT=256:SPT=16
450   GOSUB"R/W"
460 '
470 BEEP:CLS4
480   D$=USR0(CHR$(0,&H2))        :'RESTORE
490   TR0=0:FTR=0:FSCT=1:COM$="R"+CHR$(&H80):HI=&H40
500   SCT=256:SPT=16
510   GOSUB"R/W"
520   D$=USR0(CHR$(11,&H1))       :'MOTOR OFF
530 END
540 '
550 LABEL"R/W":'GIVE ME TR0,FTR,FSCT,COM$,HI,SCT,SPT
560 TR=FTR:SABA=FSCT:PLUS=SCT*SPT/256
570    FOR SIDE=0 TO 1
580      D$=USR0(CHR$(11,&H81+SIDE*16))          :'MOTOR ON
590      D$=USR0(CHR$(1,&H1E,TR,TR0)):TR0=TR     :'SEEK
600      IF HI+PLUS>=&H100 THEN P=(&H100-HI)/(SCT/256) ELSE P=SPT
610      IF SABA<>1 THEN P=P-SABA
620      D$=USR2(COM$+CHR$(SABA,P,0,HI)):SABA=1
630      PRINT ASC(D$),SABA,P,HEX$(HI)
640      HI=HI+P*(SCT/256):IF HI>=&H100 THEN RETURN
650    NEXT
660  TR=TR+1:GOTO570
670 '
680 LABEL"GSMPL"
690 CLS4:INIT
700 FOR I=0 TO 10:X1=INT(RND(1)*640):Y1=INT(RND(1)*200):C=INT(RND(1)*7)+1
710 LINE(X,Y)-(X1,Y1),XOR,C:X=X1:Y=Y1
720 NEXT:RETURN
```

リスト9-13が応用例だが，G-RAM ↔ FDの機械語ルーチン，USR2の呼び方は，

USR2 ({"R" / "W"}+CHR$（コマンド，先頭セクタ，セクタ数，アドレス下位，アドレス上位))

である。アドレスとはデータの始まるG-RAMのアドレスである。このアドレスは，グラフィック全画面のセーブ/ロードなどでは頻繁に変化するものなので，POKE文を使いたくなかったのだ。その見返りとして暴走しやすくなっているから，充分に楽しんでいただきたい。プログラムでは680行からのサブルーチンででたらめな折れ線を引き，その後フロッピーディスクにセーブ，画面クリア，G-RAMへのロードを行なっている。興味があったので，シークも含めてすべてを機械語で組んでみたところ(載せないよーだ)，理論的限界に達していることが判明した。すなわち，1周分の16セクタを読み書きするには0.2秒が必要だから，全画面をセーブ/ロードするにはシーク時間を入れなくても，どうしても2.4秒以上かかるのである。大体において，その程度の時間で読めているから，それ以上のことをするにはリスト9-10，9-11などを使って1トラック当たりの容量を増やすなどしなければならない。1セクタ1024バイトのフォーマットならば，5シリンダ（50Kバイト）で済むから，全画面ロード/セーブはシーク時間を入れずに2秒が限界になる（しかもシークが1回少なくて済む)。FDCの楽しみはこのようにして深く，広くなっていくのであった。

# TYPE Ⅳ

X1の場合はコマンドの打ち切り機能だけである。

# そしてDMAである

ここから先ではDMAが必要なので，turbo/turboZユーザーだけに意味がある。

さて，2D，2DDのディスクを扱うのならばDMAがなくとも子は育つのだが，2HDとなるとそうはいかない。データの転送速度が2倍になるので，CPUでシャコシャコとデータを送っていたならば間に合わなくなってしまうのである。そこでDMAを使ったディスクアクセスへと向かうわけである。ちなみに，リスト9-4のDMAを使わずにディスクアクセスするプログラムで試したところ，やはりしっかりと「LOST　DATAエラー」が起きてしまった。そのよーなわけであるから，やはりDMAはただ者ではないのである。

Zは別だが，turboユーザーで2HDを持っている人はあまりいないであろう。しかしご安心。DMAを使ってのディスクアクセスは，2HDばかりではないのである。すなわち，

**2D，2DDでもできるのだ。**

その場合のメリットは，

1) プログラムが短くなる
2) 恐ろしいことに，**オールBASIC**で書けてしまう

という2点である。念のために言っておくと，当たり前のことだが，2D，2DDではDMAを使っても使わなくても，ディスクの回転速度は変わらないのであるから，基本的にリード/ライトの速さは変わらない。

ところでCZ-520FやturboIII，Zの内蔵ドライブなどの2D，2DD，2HD兼用タイプのドライブを使ううえで大事な注意点がある。

それは，それらのドライブは2Dの書き込みはできるが，書き込みをしたならば他の2D専用のドライブでの読み込みは保証されないということである（2DDは問題ない)。

具体的に言うと，たとえばスイッチを切り換えて2Dモードにする。その状態で，ディスク（もちろん2Dのフォーマットになっている）に何かを書いたとする。そのディスクをそのタイプのドライブで読む分にはなんの問題もないのだが，他の2D専用のドライブ（X1G，turbo，turboIIの内蔵ドライブ，CZ-800F，801F，502Fなど）に差し替えたとすると，リードエラーが起きる可能性があるのだ。もの好きな私であるから早速試したのだが，turboの内蔵ドライブでは34番トラックあたりまではよかったのだが35番トラックでリードエラーが起きてしまった。ところがどっこいCZ-800Fではすんなりと読めてしまったのである。そのよーな状況であるから，注意していただきたい。なお，この点は他機種のドライブにおいても状況は同じだそうである。

なお，知っている人は知っていると思うが，2HDではディスクの回転速度が毎分360回（＝毎秒6回）であり，2D，2DDでは毎分300回（＝毎秒5回）である。すなわち2D，2DD，2HD兼用タイプのドライブは，動作モードによって回転速度が自動的に変わるのである。

## 脱線である

ここで一言断っておく。**今発見したところ**なのだが，turbo，turbo IIなどでは，ノーマルの2D専用ドライブで（つまり2HDを扱えないドライブで）5インチの単密度（FM方式）が使えるようなのである（つまり5インチの2Sというフォーマット)。私は今まで2Sも1Sも見たことがないし，読者もほとんどの方が見たことがないであろう（AppleIIは1Sに近いそうだが，ちょっと違う)。そのよーなわけで，以後の文中ではこのフォーマットは無視する。なにせすでに死に絶えてしまったフォーマットであるから，使い道はないのである。製造元のシャープでも2Sについては何も保証しないだろうと思われる。

## 本題である

X1の5インチ2HDの物理フォーマットにはX1フォーマットと標準フォーマットの2種類がある。X1フォーマットとは全部MFMで，77シリンダ，2サイド，各サイドには77トラックあり，1トラックには26セクタあり，セクタの容量は256バイトの（ああ面倒臭い）フォーマットである。標準フォーマットとは，IBMフォーマット（8インチ）と同じフォーマットで，第0シリンダのサイド0が単密度になっているものである。

というわけで，ここではX1フォーマットを基本として話を進める。まずは**リスト9-14**である。これはリスト9-5に相当するものである。つまり，FDCのMB8877/8876のコマンドをすべて実行できるようにしてある。BASICはturbo BASIC(CZ-8FB02)を使っていただきたい。使い方は四つの変数，TR，OLDTR，SCT，CMDに必要な値を入れてGOSUBするのである。実行に当たっては，打ち込みミスがあると悲惨なことになるので，壊れては困るファイルが入っているディスクは，すべてのドライブから抜いておくこと。また一度このプログラムを走らせた後に，LOAD，SAVEを行なうときには，まずはFILESを実行していただきたい。もしディスク周りのモードがおかしくなっていたなら，変なものが表示されるから，そのときはディスクアクセスは**すべきではない**。そして，素直にIPLをかけて，バグ捜しとなる。DMAがその筋すると電源を落とさなければならない場合もあるので心得ていただきたい。なお，1020行と1150行に「CMD=－1」という理不尽な式があるが，これはただの縁起ものと考えていただきたい。バッファ用のデータ領域は$C000_H$～を使っている。リードトラックを実行すると，$E8B0_H$あたりまで使うことになるので，まだ4Kバイトほど余裕がある。では1190行から始まるサブルーチンを順に解説する。

● “SET HD”

2HDのMFMモードにする。正確には，1200行で2HDモードにし，1210行でMFMモードにしている。2HDとはいっても，MFM（倍密度）とFM（単密度）の2種類があるのだ。

● “SET LD”

2D，2DDモードにする。

● “SET FM”

2HDのFMモード（単密度）にする。

● “MOTOR”

ドライブのモーターのON/OFF，サイドの選択をする。

● “RESTORE”

第0トラックへシークする（ヘッドを移動する）。

● “SEEK”

ヘッドを移動する。始まりのトラック番号はOLDTR，行き先のトラック番号はTRである。

● “STEP”

直前に動いた方向に，もう1トラック分シークする。

● “STEP IN”

内側（トラック番号が増える方向）に1トラック分シークする。

● “STEP OUT”

外側（トラック番号が減る方向）に1トラック分シークする。

● “READ DATA”

SCTで指定された番号のセクタを$C000_H$からに読み出す。

さて，ここで初めてDMAの登場である。DMAに対するコントロール内容は1700行に

あるとおり。すなわち，

$83_H$＝WR6 で DMA 停止の意味。

$7D_H$＝WR0 で動作は転送で，方向はポート A→ポート B を指定。次にポート A の開始アドレスとブロック長が来る。

$FB_H$，$0F_H$＝FDC のデータレジスタの I/O アドレス。

$FF_H$，$00_H$＝ブロック長(DMA の都合により－1しておく)。1セクタが 256 バイトだからこうなる。

$2C_H$＝WR1 でポート A はアドレスが固定で，I/O であることを指定。

$10_H$＝WR2でポートBはアドレスがインクリメント（増加）でメモリであることを指定。

$80_H$＝WR3 で特に何もしてない。

$8D_H$＝WR4 でバイトモードを指定。次にポート B の開始アドレスが来る。

$00_H$，$C0_H$＝メモリのデータ領域（バッファ）のアドレス。

$92_H$＝WR5 で CE/WAIT をマルチプレクス，またレディの極性は Low。

$CF_H$＝WR6 でロードコマンド。

$87_H$＝WR6 で DMA イネーブル。

以上のように DMA を設定してやると，DMA は FDC からのレディ信号を待つ状態になる。そこで FDC にリードコマンドを送ってやると，その後 FDC と DMA の間で**勝手に**データの転送をしてくれる。DMA はバイトモードであるから，転送する間を縫って CPU が動作している。データリードが終わったかどうかは，CPU が FDC のステータスレジスタを見張って，BUSY でなくなったかで判断する。終わったならば DMA に $83_H$を送って動作を停止させておく。

●“WRITE DATA”

SCT で指定された番号のセクタに $C000_H$からのデータを書き込む。

DMA に対するコントロール内容は，

$83_H$＝WR6 で DMA 停止の意味。

$79_H$＝WR0 で動作は転送で，方向はポート B→ポート A を指定(後でポート A→ポート B にひっくり返す)。次にポート A の開始アドレスとブロック長が来る。

$00_H$，$C0_H$＝データ領域のアドレス。

$FF_H$，$00_H$＝ブロック長(DMA の都合により－1しておく)。1セクタが 256 バイトだからこうなる。

$14_H$＝WR1 でポート A はアドレスがインクリメント（増加）で，メモリであることを指定。

$28_H$＝WR2 でポート B はアドレスが固定で I/O であることを指定。

$80_H$＝WR3 で特に何もしてない。

$8D_H$＝WR4 でバイトモードを指定。次にポート B の開始アドレスが来る。

$FB_H$，$0F_H$＝FDC のデータレジスタの I/O アドレス。

$92_H$＝WR5 で $\overline{\mathrm{CE}}/\overline{\mathrm{WAIT}}$ をマルチプレクス，またレディの極性は Low。

$CF_H$＝WR6 で，ロードコマンド。これにより**今のところ2バイト目の $79_H$でソース側**

**に指定されている**ポートBに開始アドレス（$0FFB_H$）がロードされる。これを怠ると正常に動作しない。

$05_H$＝WR0でポートA→ポートBに転送方向を指定し直している。

$CF_H$＝WR6でもう一度ロードコマンド。

$87_H$＝WR6でDMAイネーブル。

以上の設定の後にFDCにデータライトコマンドを送ってやると，DMAとFDCが勝手に書き込みをやってくれる。そのほかは〝READ DATA〞と同じ。

● 〝READ ID〞

ブロック長が6バイトになったこと以外は〝READ DATA〞と同じ（リスト中では－1して$05_H$，$00_H$となっている）。

● 〝READ TRACK〞

ブロック長が$28B0_H$バイトになったこと以外は〝READ DATA〞と同じ。

● 〝WRITE TRACK〞

ブロック長が$28B0_H$バイトになったこと以外は〝WRITE DATA〞と同じ。

● 〝FORCE INT〞

FDCにコマンドを送るだけ。

このよーにしてDMAを使ったディスクアクセスはオールBASICしてしまうのであった。リスト 9-14 ではCZ-520Fをドライブ2，3とした場合を考えている。1020行をGOSUB 〝SET LD〞にし，1030行などで，〝MOTOR〞に与えるコマンド（CMD）を&H81などにすると内蔵のドライブ1にDMAを使ってアクセスできる。

**リスト 9-14　DMAでディスクアクセス**

```
1000 CLEAR &HC000
1010 'TR,OLDTR,SCT,CMD
1020                     CMD=-1   :GOSUB"SET HD"      :GOSUB"ER?"
1030                     CMD=&H82:GOSUB"MOTOR"       :GOSUB"ER?"
1040                     CMD=&H0  :GOSUB"RESTORE"     :GOSUB"ER?"
1050 'TR=1 :OLDTR=0 :CMD=&H1C:GOSUB"SEEK"         :GOSUB"ER?"
1060 '                   CMD=&H38:GOSUB"STEP"         :GOSUB"ER?"
1070 '                   CMD=&H58:GOSUB"STEP IN"      :GOSUB"ER?"
1080 '                   CMD=&H78:GOSUB"STEP OUT"     :GOSUB"ER?"
1090  SCT=1             :CMD=&H80:GOSUB"READ DATA"   :GOSUB"ER?"
1100  SCT=1             :CMD=&HA0:GOSUB"WRITE DATA"  :GOSUB"ER?"
1110 '                   CMD=&HC0:GOSUB"READ ID"      :GOSUB"ER?"
1120 '                   CMD=&HE0:GOSUB"READ TRACK"   :GOSUB"ER?"
1130 '                   CMD=&HF0:GOSUB"WRITE TRACK":GOSUB"ER?"
1140 '                   CMD=&HD0:GOSUB"FORCE INT"    :GOSUB"ER?"
1150                     CMD=-1   :GOSUB"SET LD"       :GOSUB"ER?"
1160                     CMD=&H2  :GOSUB"MOTOR"        :GOSUB"ER?"
1170 END
1180 '
1190 LABEL"SET HD"
1200   DUMMY=INP(&HFFE)   :'2HD MODE
1210   DUMMY=INP(&HFFD)   :'MFM MODE
1220   STAT=0
1230 RETURN
1240 '
1250 LABEL"SET LD"
1260   DUMMY=INP(&HFFF)   :'2D/2DD MODE
1270   DUMMY=INP(&HFFD)   :'MFM MODE
1280   STAT=0
1290 RETURN
1300 '
1310 LABEL"SET FM"
1320   DUMMY=INP(&HFFE)   :'2HD MODE
```

```
1330   DUMMY=INP(&HFFC)  :'FM MODE
1340   STAT=0
1350 RETURN
1360 '
1370 LABEL"MOTOR"
1380   OUT &HFFC,CMD
1390   IF CMD AND &H80 THEN GOSUB"WNBSY"
1400 RETURN
1410 '
1420 LABEL"RESTORE"
1430   OUT &HFF8,CMD
1440   GOSUB"WNBSY"
1450 RETURN
1460 '
1470 LABEL"SEEK"
1480   OUT &HFFB,TR       :'DATA REG.（目的のシリンダ番号）
1490   OUT &HFF9,OLDTR    :'TRACK REG.（現在のシリンダ番号）
1500   OUT &HFF8,CMD
1510   GOSUB"WNBSY"
1520 RETURN
1530 '
1540 LABEL"STEP"
1550   OUT &HFF8,CMD
1560   GOSUB"WNBSY"
1570 RETURN
1580 '
1590 LABEL"STEP IN"
1600   OUT &HFF8,CMD
1610   GOSUB"WNBSY"
1620 RETURN
1630 '
1640 LABEL"STEP OUT"
1650   OUT &HFF8,CMD
1660   GOSUB"WNBSY"
1670 RETURN
1680 '
1690 LABEL"READ DATA"
1700   DMA$=HEXCHR$("83 7D FB 0F FF 00 2C 10 80 8D 00 C0 92 CF 87")
1710   GOSUB"SETDMA"
1720   OUT &HFFA,SCT      :'SECTOR REG.
1730   OUT &HFF8,CMD
1740   GOSUB"WNBSY"
1750   GOSUB"RESETDMA"
1760 RETURN
1770  '
1780 LABEL"WRITE DATA"
1790   DMA$=HEXCHR$("83 79 00 C0 FF 00 14 28 80 8D FB 0F 92 CF 05 CF 87")
1800   GOSUB"SETDMA"
1810   OUT &HFF8,CMD
1820   GOSUB"WNBSY"
1830   GOSUB"RESETDMA"
1840 RETURN
1850 '
1860 LABEL"READ ID"
1870   DMA$=HEXCHR$("83 7D FB 0F 05 00 2C 10 80 8D 00 C0 92 CF 87")
1880   GOSUB"SETDMA"
1890   OUT &HFF8,CMD
1900   GOSUB"WNBSY"
1910   GOSUB"RESETDMA"
1920 RETURN
1930  '
1940 LABEL"READ TRACK"
1950   DMA$=HEXCHR$("83 7D FB 0F AF 28 2C 10 80 8D 00 C0 92 CF 87")
1960   GOSUB"SETDMA"
1970   OUT &HFF8,CMD
1980   GOSUB"WNBSY"
1990   GOSUB"RESETDMA"
2000 RETURN
2010  '
2020 LABEL"WRITE TRACK"
2030   DMA$=HEXCHR$("83 79 00 C0 AF 28 14 28 80 8D FB 0F 92 CF 05 CF 87")
2040   GOSUB"SETDMA"
2050   OUT &HFF8,CMD
2060   GOSUB"WNBSY"
2070   GOSUB"RESETDMA"
2080 RETURN
2090 '
2100 LABEL"FORCE INT"
2110   OUT &HFF8,CMD
2120   STAT=INP(&HFF8)
2130 RETURN
2140 '
```

```
2150 LABEL"RESETDMA"
2160   DMA$=HEXCHR$("83"):GOSUB"SETDMA"
2170 RETURN
2180 '
2190 LABEL"WNBSY"
2200   CT=0
2210   IF CT>1000 THEN OUT &HFFC,&H3 AND DN:PRINT"DISK?":STOP
2220   STAT=INP(&HFF8):IF STAT AND &H81 THEN CT=CT+1:GOTO 2210
2230 RETURN
2240 '
2250 LABEL"SETDMA"
2260   FOR I=1 TO LEN(DMA$):OUT &H1F80,ASC(MID$(DMA$,I,1)):NEXT
2270 RETURN
2280 '
2290 LABEL"ER?"
2300   PRINTHEX$(CMD),HEX$(STAT)
2310 RETURN
```

## 2HDの物理フォーマットである

DMAを使ったディスクフォーマットプログラムであるが，リスト9-9を**図9-11**に従って削除/訂正/追加していただきたい(1010～1170行の削除を忘れずに)。できあがったものがリスト9-8に相当するディスクフォーマットプログラムである。これはturbo BASICでも動くようになっている(もちろんCZ-8FB01でも動く)。RUNするとドライブ番号1(2HD)に入っているディスクに物理フォーマットをかけるようにしてある。打ち込みミスがあると悲惨なことになるので，壊れては困るファイルが入っているディスクはすべてのドライブから抜いておくこと。

さらには2000行以降を**リスト9-15**と差し替えると，1セクタの容量が512バイトでセ

図9-11 「リスト9-9」→DMAを使った2HDディスクフォーマットプログラムへの変更点

```
1010～1170行を削除

1200 AD=&HC000            :'フォーマット用データのアドレス

1230 DUMMY=INP(&HFFE)                        :'2HDモードにする

1290 FOR CYLINDER=0 TO 76                    :'シリンダ数変更

1440 DUMMY=INP(&HFFE)                        :'2HDモードにする

1650 'D2$=USR0(CHR$(9,&HF0,1)):STAT=ASC(D2$):RETURN      :'削除する

2000 LABEL"GAP1"
2010 DATA 80,&H4E,12,&H00,3,&HF6,1,&HFC,50,&H4E,0
2020 LABEL"SECTOR"
2030 DATA 12,0,3,&HF5,1,&HFE,4,!
2040 DATA 1,&HF7,22,&H4E,12,0,3,&HF5,1,&HFB,256,&HE5,1,&HF7,54,&H4E,0
2050 LABEL"GAP4"      :'                      ^              ^
2060 DATA 598,&H4E,0  :'<-!!!                !!!            !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 26          :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
2110 DATA 17,18,19,20,21,22,23,24,25,26
2120 LABEL"TYPE"
2130 DATA 1           :'<-!!! 1 SECTOR 256 Byte つぅことでんがな
```

クタの数が15個，**リスト9-16**と差し替えると1セクタの容量が1024バイトでセクタの数が8個のフォーマットになる。挙げ句の果てに，**リスト9-17**はFM（単密度）用のフォーマットである。1235行と1445行もいっしょに付け加えていただきたい。

もちろん少しの訂正で2D，2DDにフォーマットをかけるように変更することも可能である。その点については自由研究である。

**リスト9-15　2HD　1セクタ512バイトへの変更点**

```
2000 LABEL"GAP1"
2010 DATA 80,&H4E,12,&H00,3,&HF6,1,&HFC,50,&H4E,0
2020 LABEL"SECTOR"
2030 DATA 12,0,3,&HF5,1,&HFE,4,!
2040 DATA 1,&HF7,22,&H4E,12,0,3,&HF5,1,&HFB,512,&HE5,1,&HF7,84,&H4E,0
2050 LABEL"GAP4"      :'                       ^              ^
2060 DATA 400,&H4E,0  :'<-!!!                 !!!            !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 15         :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
2110 LABEL"TYPE"
2120 DATA 2          :'<-!!! 1 SECTOR 512 Byte つぅことでんがな
```

**リスト9-16　2HD　1セクタ1024バイトへの変更点**

```
2000 LABEL"GAP1"
2010 DATA 80,&H4E,12,&H00,3,&HF6,1,&HFC,50,&H4E,0
2020 LABEL"SECTOR"
2030 DATA 12,0,3,&HF5,1,&HFE,4,!
2040 DATA 1,&HF7,22,&H4E,12,0,3,&HF5,1,&HFB,1024,&HE5,1,&HF7,116,&H4E,0
2050 LABEL"GAP4"      :'                        ^               ^
2060 DATA 654,&H4E,0  :'<-!!!                  !!!             !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 8          :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5,6,7,8
2110 LABEL"TYPE"
2120 DATA 3          :'<-!!! 1 SECTOR 1024 Byte つぅことでんがな
```

**リスト9-17　2HD　1セクタ128バイトへの変更点**

```
1235 DUMMY=INP(&HFFC)                    :'ＦＭモードにする

1445 DUMMY=INP(&HFFD)                    :'ＭＦＭモードにする

2000 LABEL"GAP1"
2010 DATA 40,&HFE,6,&H00,1,&HFC,26,&HFF,0
2020 LABEL"SECTOR"
2030 DATA 6,&H00,1,&HFE,4,!
2040 DATA 1,&HF7,11,&HFF,6,&H00,1,&HFB,128,&HE5,1,&HF7,27,&HFF,0
2050 LABEL"GAP4"      :'                       ^              ^
2060 DATA 247,&HFF,0  :'<-!!!                 !!!            !!!
2070 LABEL"TOTAL SECTOR"
2080 DATA 26        :'<-!!!
2090 LABEL"SQUE"
2100 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
2110 DATA 17,18,19,20,21,22,23,24,25,26
2120 LABEL"TYPE"
2130 DATA 0          :'<-!!! 1 SECTOR 128 Byte つぅことでんがな
```

以上のフォーマット用のデータの内訳は，例によって参考文献10のお世話になっている。

## グラフィックするのである

DMAの性質としてメモリとI/Oは同等に扱えるということである。そこで私の傾向と性格により，リスト9-14に**リスト9-18**を追加変更するとたちまちにしてFD↔G-RAMプログラムができてしまうのである。変更点は，ディスクからリードする場合は転送先をI/Oにしたことと，アドレスをG-RAMの始まりの$4000_H$にしたことである。ライトする場合はよって知るべしである。———と書くともっともらしく聞こえるが，実はそれだけでは動かない。というのは，BASICがG-RAMのバンク0，1を勝手に切り換えるからなのだ。実に残念なことよ。

**リスト9-18　G-RAMアクセス用変更点**

```
1700    DMA$=HEXCHR$("83 7D FB 0F FF 00 2C 18 80 8D")
1705    DMA$=DMA$+MKI$(&H4000)+HEXCHR$("92 CF 87")

1790    DMA$=HEXCHR$("83 79")+MKI$(&H4000)
1795    DMA$=DMA$+HEXCHR$("FF 00 1C 28 80 8D FB 0F 92 CF 05 CF 87")
```

## ソフト的なフォーマットである

2HD, 2DDのディスクでは容量が違うのであるから，BASICによるファイル管理も少々違っている。詳しくはturboのUSER'S MANUALの187ページあたりに書いてあるが，大事な点は2HDでも1クラスタは4Kバイトのままで，これは1トラック(片面)の容量ではないということである。2Dのなごりを引きずっているわけだ。さらにはクラスタ番号である。2DD, 2HDではクラスタ数がそれぞれ128以上あるが，FAT内では$80_H$〜$8F_H$はチェーンの終わりを示すものであったから，この番号のクラスタはあってはいけないことになっている。そこでBASICでどのようにごまかしているかというと，

1) 実はクラスタ番号は2バイトの値（16進数で4桁）を持つ
2) そして，**下2桁が$80_H$〜$FF_H$の部分は番号を飛ばす**

となっているのである。要するに，2DD, 2HD(8インチやハードディスクも)ではFATの位置（レコード番号）やディレクトリの位置が違うということである。

## さらにグラフィックするのである

**リスト9-19**である。このプログラムはリスト9-14＋リスト9-16で作った，1セクタが1024バイトのフォーマットのディスクを画像用にしてしまうものである。具体的に言うと，1トラック（片面）に8セクタあり，1シリンダ（裏表で計2トラック）が16Kバイ

トであることから，3シリンダに 48K バイト分のグラフィックデータを書き込んでしまうのである。ディスクアクセスにはマルチセクタを使っている。FDC の説明の最初の所でも書いたが，マルチセクタを使うと FDC は「RECORD NOT FOUND」でエラーを起こすまで BUSY のままである。そこでここでは DMA の「エンド・オブ・ブロックで割り込み」の機能を使っている。これにより FDC が第9セクタ（ここでは存在しない）を捜してエラーを起こすのを待たずに済むようになっている。最初にある MEM$は，FDC のステータスレジスタを読み出すルーチン＋割り込み処理ルーチンである。「FDC のステータスをチェックするなら BASIC の INP 関数を使えばよい」と思うだろうが，実はそこには恐ろしいワナが隠されている。なんと，私の調べたところによると，INP 関数を使うと BASIC は，

**勝手に G-RAM の入力ページ（バンク 0，1）をひっくり返してしまう**

のだ。よって CALL 命令で呼び出した機械語プログラムによって FDC のステータスをチェックしなければならないのだ。もっともこの部分は徹底的に探ったわけではないから，いまいち確信はないのだが。**ま，とにかく**そのようなわけなのである。

リスト 9-19　48K　G-RAM ロード/セーブプログラム

```
100 CLEAR &HEF00
110 MEM$(&HEF00,16)=HEXCHR$("01 F8 0F ED 78 32 26 EF C9 F3 F5 C5 01 80 1F 3E")
120 MEM$(&HEF10,16)=HEXCHR$("AF ED 79 3E 8B ED 79 3E AB ED 79 3E 01 32 25 EF")
130 MEM$(&HEF20, 7)=HEXCHR$("C1 F1 FB ED 4D 00 00")
140 INIT
150 DN=1
160 DUMMY=INP(&HFFE)              :'2HD(1.6) MODE
170 MEM$(&HF814,2)=MKI$(&HEF09) :'OLD VALUE=&H81C3
180 '
190 GOSUB"RESTORE":OLDTR=0
200 FOR TTR=0 TO 72 STEP 3
210   CLS4
220   SYMBOL(32,0),STR$(TTR),20,10,7,0,PSET
230   GOSUB"WRITE"
240 NEXT
250 '
260 GOSUB"RESTORE":OLDTR=0
270 FOR TTR=0 TO 72 STEP 3
280   CLS4
290   GOSUB"READ"
300 NEXT
310 '
320 OUT &HFFC,&H3 AND DN
330 DUMMY=INP(&HFFF):'2D,2DD(0.5M,1M) MODE
340 MEM$(&HF814,2)=MKI$(&H81C3)
350 END
360 '
370 LABEL"READ"
380 CCC=&H90
390 DD1$=HEXCHR$("83 7D FB 0F FF 1F 2C 18 9D")
400 DD2$=HEXCHR$("32 10 92 A0 CF 8B 87")
410 GOSUB"EXEC"
420 RETURN
430 '
440 LABEL"WRITE"
450 CCC=&HB0
460 DD1$=HEXCHR$("83 79")
470 DD2$=HEXCHR$("FF 1F 1C 28 9D FB 0F 32 10 92 CF 05 A0 CF 87")
480 GOSUB"EXEC"
490 RETURN
500 '
510 LABEL"EXEC"
520 GAD=&H4000
530 FOR TR=TTR TO TTR+2
540   GOSUB"SEEK":OLDTR=TR
```

```
550   FOR SIDE=0 TO 1
560     DMA$=DD1$+MKI$(GAD)+DD2$
570     GOSUB"SETDMA"
580     OUT &HFFC,&H80 OR DN OR &H10*SIDE:GOSUB "WNBSY"  :'MOTOR ON
590     OUT &HFFA,1                                     :'SECTOR #
600     POKE &HEF25,0   :'CLEAR FLAG
610     OUT &HFF8,CCC                                   :'COMMAND
620     CALL &HEF00
630     IF (PEEK(&HEF26) AND 1)=0 THEN 660
640     IF PEEK(&HEF25) THEN 660
650     GOTO 620
660     PRINT PEEK(&HEF25),PEEK(&HEF26),INP(&HFF8)
670     OUT &HFF8,&HD0
680     GAD=GAD+&H2000
690   NEXT
700 NEXT
710 DMA$=HEXCHR$("C3 C3 C3 C3 C3 C3 83 80"):GOSUB"SETDMA"
720 RETURN
730 '
740 LABEL"WNBSY"
750 CT=0
760 IF CT>1000 THEN OUT &HFFC,&H2:PRINT"DISK?":STOP
770 IF INP(&HFF8) AND &H81 THEN CT=CT+1:GOTO 760
780 RETURN
790 '
800 LABEL"SETDMA"
810 FORI=1TOLEN(DMA$):OUT &H1F80,ASC(MID$(DMA$,I,1)):NEXT
820 RETURN
830 '
840 LABEL"SEEK"
850 OUT &HFFB,TR          :'データレジスタ
860 OUT &HFF9,OLDTR       :'トラックレジスタ
870 OUT &HFF8,&H10        :'シークコマンド
880 GOSUB"WNBSY"
890 RETURN
900 '
910 LABEL"RESTORE"
920 OUT &HFFC,&H80 OR DN:GOSUB "WNBSY":'MOTOR ON
930 OUT &HFF8,&H0 :GOSUB "WNBSY":'HOME
940 RETURN
```

それでリスト 9-19 でやっていることだが，中心は 370 行以降の〝READ〟と〝WRITE〟である。それぞれのルーチンは変数 TTR（Top Track）から始まる 3 シリンダと 48K バイトの G-RAM の間でデータの読み書きを行なう。プログラム全体では，最初は SYMBOL 文を使って G-RAM に TTR の値をでかでかと書き，それを次々とディスクに書き込み，次には，次々に読み出して表示している。これは実行させてみれば一目瞭然であろう。そこで 110～130 行の MEM$～による機械語プログラムを簡単に説明しておく。アセンブルリストは，**リスト 9-20** である。EF00$_H$～EF08$_H$は単に FDC のステータスを読み出して EF26$_H$番地に格納しているだけである。大事なのは EF09$_H$～で，これは「エンド・オブ・ブロック」後の DMA の再初期化と「フラグ立て」(EF25$_H$番地）を行なっている。このフラグが立ったなら，DMA が 8K バイトの転送を終えたわけだから，FDC の空しい第 9 セクタ捜しをフォースインタラプトで打ち切ってやるわけである。

**リスト 9-20 「リスト 9-19」の機械語部分**

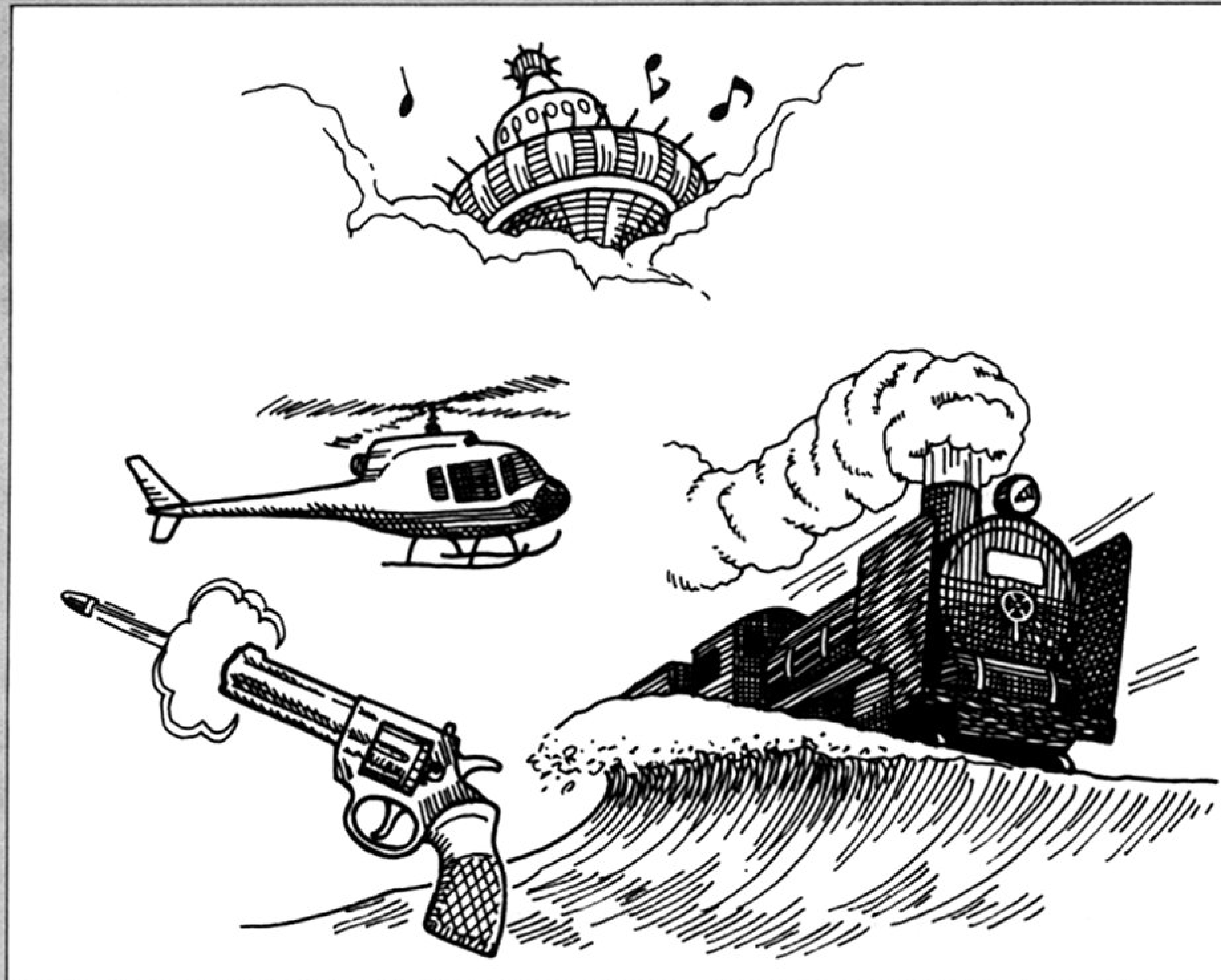
```
                                      .Z80
                                      .PHASE   0EF00H
                              ;
0FF8                          FDC     EQU      0FF8H     ;FDC CMD AND STAT
1F80                          DMA     EQU      1F80H     ;DMA I/O ADDR.
                              ;
EF00     01 0FF8              START:  LD       BC,FDC
EF03     ED 78                        IN       A,(C)
EF05     32 EF26                      LD       (STAT),A
```

```
EF08     C9                         RET
                            ;
EF09     F3                 DMAINT: DI
EF0A     F5                         PUSH    AF
EF0B     C5                         PUSH    BC
EF0C     01 1F80                    LD      BC,DMA
EF0F     3E AF                      LD      A,0AFH  ;WR6:DISABLE INT.
EF11     ED 79                      OUT     (C),A
EF13     3E 8B                      LD      A,08BH  ;WR6:REINIT. STAT
EF15     ED 79                      OUT     (C),A
EF17     3E AB                      LD      A,0ABH  ;WR6:ENABLE INT.
EF19     ED 79                      OUT     (C),A
EF1B     3E 01                      LD      A,1
EF1D     32 EF25                    LD      (INTF),A;SET FLAG
EF20     C1                         POP     BC
EF21     F1                         POP     AF
EF22     FB                         EI
EF23     ED 4D                      RETI
                            ;
EF25                        INTF:   DS      1
EF26                        STAT:   DS      1
                            ;
                                    END
```

以上がDMAでディスクを使ってみた例である。残念ながら8インチやハードディスクなどは手付かずとなってしまったが，さっさと次の章に行ってしまうのであった。

# 第10章

# PSG



PSGは基本である

## 第10章
# PSGは基本である・・・・・・・・・・・・

この章ではPSGをやるのである。PSGというのは，Programmable Sound Generator，すなわち「プログラム可能な音声発生機」なのである。プログラム可能とはいっても**別にたいしたことはない**からそのつもりで。

X1で使われているPSGはAY-3-8910という，もう古典と言ってもよいLSIである。ただしturboZなどでは，それとコンパチなYAMAHAの YM2149も使われている(ちなみにYM2149は「SSG」と呼ばれている。このことから逆にAY-3-8910をSSGと呼んだりもする)。

で，この石を簡単に紹介すると，

**1)** 音量調節付きで3重和音が使える。

**2)** ノイズ（雑音）を使える。

**3)** エンベロープと呼ばれる機能により，指定した周期，形状（8パターン）で音量を変化させることができる。

**4)** 2ポートの8ビットパラレルポートが使える。X1ではこれをジョイスティックに使っている。

などとなる。

AY-3-8910から**4)**の機能を取り去ってピン数を減らしたものは型番がAY-3-8913と呼ばれ，MZ-5500/6500で使われている。FM-7では8910，8913が混在しているそうである。どっちにしても，FM-7ではパラレルポートを使ってジョイスティックをサポートしていないから同じことである。

AY-3-8910の使い方は簡単で，基本的には16個あるレジスタにそれなりのデータを書き込んでやったり，もしくは読み出したりすればよい。この点については第1章のCRTCと似ている。そこで早速PSGのレジスタ表が**表10-1**である。

最初に言っておくが，$R_0$と$R_1$，$R_2$と$R_3$，$R_4$と$R_5$は別々に見ずに「12ビットの値を下8ビットと上4ビットに分けた」と見るべきである。これは$R_{11}$と$R_{12}$でも同じで，こちらの方は16ビットの値を上下8ビットずつに分けたものとみなすことができる。この点を注意しつつ，順に解説する。

### ① $R_0$，$R_1$（チャンネルAトーン周波数）

$R_1$を上4ビット，$R_0$を下8ビットとして作る12ビットの整数を$A_f$とする。$A_f$は0〜4095である。すなわち，

$A_f = R_0 + R_1 \times 256$……**式1**

この$A_f$とチャンネルAから出る音の周波数の関係は，音の周波数を$f_a$(Hz)とすると，

$f_a = \dfrac{2\times10^6}{16\times A_f} = \dfrac{125000}{A_f}$……**式2**

表 10-1　PSG レジスタ表

| レジスタ番号 | レジスタ機能 | ビット構成 D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | データ |
|---|---|---|---|---|---|---|---|---|---|---|
| $R_0$ | チャンネル A 周波数 | 下位 8 ビット | | | | | | | | 0 ~ 255 |
| $R_1$ | | | | | | 上位 4 ビット | | | | 0 ~ 15 |
| $R_2$ | チャンネル B 周波数 | 下位 8 ビット | | | | | | | | 0 ~ 255 |
| $R_3$ | | | | | | 上位 4 ビット | | | | 0 ~ 15 |
| $R_4$ | チャンネル C 周波数 | 下位 8 ビット | | | | | | | | 0 ~ 255 |
| $R_5$ | | | | | | 上位 4ビット | | | | 0 ~ 15 |
| $R_6$ | ノイズ周波数 | | | | 5 ビット | | | | | 0 ~ 31 |
| $R_7$ | チャンネル選択 | $\overline{\text{IN}}$/OUT | | $\overline{\text{ノイズ}}$ | | | $\overline{\text{トーン}}$ | | | 0~255 |
| | | IOB | IOA | C | B | A | C | B | A | ($D_0$~$D_5$ は，0 を設定すると選択される) |
| $R_8$ | チャンネル A 音量 | | | | M | 4 ビット | | | | 0 ~ 16 |
| $R_9$ | チャンネル B 音量 | | | | M | 4 ビット | | | | 0 ~ 16 |
| $R_{10}$ | チャンネル C 音量 | | | | M | 4 ビット | | | | 0 ~ 16 |
| $R_{11}$ | エンベロープ周期 | 下位 8 ビット | | | | | | | | 0 ~ 255 |
| $R_{12}$ | | 上位 8 ビット | | | | | | | | 0 ~ 255 |
| $R_{13}$ | エンベロープ形状 | | | | | 4 ビット | | | | 0 ~ 15 |
| $R_{14}$ | JOY 1 (IOA) | 8 ビット | | | | | | | | 0 ~ 255 |
| $R_{15}$ | JOY 2 (IOB) | 8 ビット | | | | | | | | 0 ~ 255 |

となっている。**式 2** で「$2\times10^6$」と「16」という数値がいきなり出てくるが，$2\times10^6$ というのは，LSI に供給されているクロックの周波数（2MHz）である。だからハードウェアが異なれば，この部分の値は変わってくる。M（メガ）というのは $10^6=1000000$ の単位なのだ，念のため。 次に出てくる 16 は，AY-3-8910 を設計した人が勝手に決めた数値である（と思う）。なお，**式 2** より $A_f$ の値が大きいほど周波数の小さい音（低い音）が出ることが分かる。要するに逆比例なのだ。

**式 1，2** より「周波数 f（Hz）の音を出したいときに，どういう数値を $R_0$，$R_1$ に書き込めばよいか」を計算できる。

まず，**式 2** から，

$$A_f=\frac{125000}{f_a}$$

が分かる。ただし $A_f$ は整数じゃないとまずいので，小数点第 1 位で四捨五入するのがよいだろう。そこで，本当は，

$$A_f=\mathrm{INT}\left(\frac{125000}{f_a}+0.5\right)$$

となる。次に $A_f$ から $R_0$，$R_1$ を計算するわけだが，これは簡単に，

$$R_0=A_f \bmod 256$$

$$R_1=\mathrm{INT}\left(\frac{A_f}{256}\right)$$

となる。めでたしめでたし。ただし今のうちに言っておくが，$R_0$，$R_1$だけをセットしても音は出てこない。注意するように。なお，やってみると分かるが，$A_f$を0にした場合，式の上では周波数が無限大の音が出ることになる。自然界では当然そんなことはあり得ないので，$A_f=0$の場合は考える必要はない。また，音質を良くするためにも電気回路では，あまり高い周波数（成分）の音は増幅されないのが普通だから，そもそもスピーカーから出ていないはずである。

さてさて，音の高さについて書いてきたのであるが，周波数ばかりで説明していても始まらない。実用的にはドレミの音階が必要である。そこで**表10-2**が，えーとえーと，「1939年5月ロンドンにおける国際会議で規定された音階表」である。これは『理科年表』に載っていたものである。

**表10-2　国際基準イ＝$a^1$＝440Hzに基づく十二平均律音階**　（単位はHz）

| | $C_2$ | $C_1$ | C | c | $c^1$ | $c^2$ | $c^3$ | $c^4$ | $c^5$ |
|---|---|---|---|---|---|---|---|---|---|
| C | 16.352 | 32.703 | 65.406 | 130.81 | 261.63 | 523.25 | 1046.5 | 2093.0 | 4186.0 |
| C ♯ | 17.324 | 34.648 | 69.296 | 138.59 | 277.18 | 554.37 | 1108.7 | 2217.5 | 4434.9 |
| D | 18.354 | 36.708 | 73.416 | 146.83 | 293.66 | 587.33 | 1174.7 | 2349.3 | 4698.6 |
| D ♯ | 19.445 | 38.891 | 77.782 | 155.56 | 311.13 | 622.25 | 1244.5 | 2489.0 | 4978.0 |
| E | 20.602 | 41.203 | 82.407 | 164.81 | 329.63 | 659.26 | 1318.5 | 2637.0 | 5274.0 |
| F | 21.827 | 43.654 | 87.307 | 174.61 | 349.23 | 698.46 | 1396.9 | 2793.8 | 5587.7 |
| F ♯ | 23.125 | 46.249 | 92.499 | 185.00 | 369.99 | 739.99 | 1480.0 | 2960.0 | 5919.9 |
| G | 24.500 | 48.999 | 97.999 | 196.00 | 392.00 | 783.99 | 1568.0 | 3136.0 | 6271.9 |
| G ♯ | 25.957 | 51.913 | 103.83 | 207.65 | 415.30 | 830.61 | 1661.2 | 3322.4 | 6644.9 |
| A | 27.500 | 55.000 | 110.00 | 220.00 | 440.00 | 880.00 | 1760.0 | 3520.0 | 7040.0 |
| A ♯ | 29.135 | 58.270 | 116.54 | 233.08 | 466.16 | 932.33 | 1864.7 | 3729.3 | 7458.6 |
| B | 30.868 | 61.735 | 123.47 | 246.94 | 493.88 | 987.77 | 1975.5 | 3951.1 | 7902.1 |

（1979年度版理科年表より）

この表には厳格な規則性がある。中央に四角で囲んだ数字があるが，これが$a^1$＝イと呼ばれる音で，何がなんでもここは440Hzなのである。これはラの音である。X1で出すのなら，

PLAY "O4A"

で出てくる。この音を基準として，半音上げた音（♯A）の周波数は，440に$\sqrt[12]{2}=2^{\frac{1}{12}}=1.05946......$（2の12乗根）を掛けたもの，逆に半音下げた音（♯G）は440を$\sqrt[12]{2}$で割ったものである。同じように$\sqrt[12]{2}$で掛けたり割ったりすると，そのたびに半音上がったり下がったりする。1オクターブは半音が12個分だから，1オクターブ上の音は，ちょうど周波数が2倍になる（$\sqrt[12]{2}$の12乗は2なのだ）。逆に1オクターブ下の音は，周波数がちょうど2分の1である。表10-2を見て納得していただきたい。なお，なぜこのように決められているのか不思議に思う人もいるだろうが，それは人間の聴覚システムがそうなっているからなのである。それ以上のことは私の知ったことではない。

では，AY-3-8910で音階表どおりの周波数の音を出すにはどうすればよいかという問題に移る。真面目に計算してもよいのだが，私はturbo BASICのMUSIC@命令を使ったのである。MUSIC命令だと，音が出ている間はずっと待たされてしまうが，MUSIC@命令

ならば待たされることなく次の命令を実行してくれる。だからそのときにPSGの$R_0$と$R_1$を読み出せばよい。なお，それじゃMUSIC@命令はいつ音を止めるのかというと，タイマ割り込みを使っているのである。だから心配せずとも，時期が来れば止まってくれる。

**表10-3**が$R_0$と$R_1$の値を表にしたものである。16進数で，「$R_0+R_1$」の形式でできている。四角で囲んであるところが440 Hzに対応する所である。とはいっても，本当に正確に440 Hzを出せるわけではない。なにせ$R_0$，$R_1$は整数に限られるのである。これはほかの音階でも言えることで，特に高い音になるほどずれは大きくなる。まあ，こういうものだと思うしかないのである。なお，1オクターブ上げる＝周波数を2倍する＝$A_f$を半分にする，であることに注意。

表10-3　各音階に対するレジスタの値(「$R_0$」+「$R_1$」の形式)

| | C | #C | D | #D | E | F | #F | G | #G | A | #A | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| O1 | EF+0E | 17+0E | 4D+0D | 8E+0C | DA+0B | 2F+0B | 8F+0A | F7+09 | 68+09 | E1+08 | 62+08 | E9+07 |
| O2 | 77+07 | 0B+07 | A6+06 | 47+06 | ED+05 | 97+05 | 47+05 | FB+04 | B4+04 | 70+04 | 31+04 | F4+03 |
| O3 | BB+03 | 85+03 | 53+03 | 23+03 | F6+02 | CB+02 | A3+02 | 7D+02 | 5A+02 | 38+02 | 18+02 | FA+01 |
| O4 | DD+01 | C2+01 | A9+01 | 91+01 | 7B+01 | 65+01 | 51+01 | 3E+01 | 2D+01 | 1C+01 | 0C+01 | FD+00 |
| O5 | EE+00 | E1+00 | D4+00 | C8+00 | BD+00 | B2+00 | A8+00 | 9F+00 | 96+00 | 8E+00 | 86+00 | 7E+00 |
| O6 | 77+00 | 70+00 | 6A+00 | 64+00 | 5E+00 | 59+00 | 54+00 | 4F+00 | 4B+00 | 47+00 | 43+00 | 3F+00 |
| O7 | 3B+00 | 38+00 | 35+00 | 32+00 | 2F+00 | 2C+00 | 2A+00 | 27+00 | 25+00 | 23+00 | 21+00 | 1F+00 |
| O8 | 1D+00 | 1C+00 | 1A+00 | 19+00 | 17+00 | 16+00 | 15+00 | 13+00 | 12+00 | 11+00 | 10+00 | 0F+00 |

**② $R_2$，$R_3$（チャンネルBトーン周波数）**

**③ $R_4$，$R_5$（チャンネルCトーン周波数）**

②，③はチャンネルAがB，Cになっただけで，それ以外はまったく同じである。

**④ $R_6$（ノイズ周波数）**

AY-3-8910には前述したようにノイズを出す機能がある。といっても別にノイズ専門のチャンネルがあるわけではなく，チャンネルA〜Cのうちから割り当てるのである。割り当て方については⑤で説明する。

ノイズとはどんなときに使うのかといえば，波の音とか爆発音などの効果音ということになるだろう。$R_6$はそのノイズの周波数を決めるものである。だが，ノイズというのは，一定の幅の周波数帯の音を含んでいる音のことだから，大体の範囲を決めるということである。具体的には，周波数が高ければ「シャー」，低ければ「ザー」という感じの音になる。$R_6$とノイズの高さは，例によって逆比例の関係になる。つまり，$R_6$の値が小さいほどノイズの音は高くなる。

**⑤ $R_7$（ミキサー，I/Oコントロール）**

この部分はフラグの集まりと考えるのが手っ取り早い。まず，$D_0$〜$D_5$はチャンネルA，B，Cから何を出すのかの指定である。チャンネルAを指定するのは$D_3$と$D_0$である。**図10-1**に$D_3$と$D_0$の値（1/0）とチャンネルAからの出力の関係を示す。

図10-1はほかのチャンネルにとっても同じことが言える。1でOFF，0でONと変態的に負論理になっていることに注意。

$D_6$，$D_7$は二つあるI/Oポートを入力に使うか出力に使うかの指定である。もちろん二つとも別々に指定することができる。0で入力に指定，1で出力に指定である。X1では通常は二つともジョイスティックからの入力になっているから，ともに0である。なお，実際

図 10-1　ノイズ/トーン指定ビット
（チャンネル A の場合）

| $D_3$ | $D_0$ | チャンネル A 出力 |
|---|---|---|
| 0 | 0 | ノイズ＋トーン |
| 0 | 1 | ノイズ |
| 1 | 0 | トーン |
| 1 | 1 | なし |

のジョイスティックポートのデータの受け渡しは $R_{14}$，$R_{15}$を使う。

**⑥　$R_8$，$R_9$，$R_{10}$（チャンネル A〜C 音量，エンベロープ指定）**

$R_8$，$R_9$，$R_{10}$のそれぞれ下 4 ビットの $D_3$〜$D_0$は音量を指定する。4 ビットだから 0（最小）〜15（最大）までの 16 段階である。ただし $D_4$が 1 ならば，そのチャンネルはエンベロープモードになり，$D_3$〜$D_0$の値は無視される。この場合の音量は，$R_{11}$，$R_{12}$，$R_{13}$の指定するエンベロープ周期と，エンベロープパターンに支配される。

**⑦　$R_{11}$，$R_{12}$（エンベロープ周期設定）**

これは，次で指定するエンベロープのパターンの周期を設定するものである。エンベロープというのは，**図 10-2** にあるように 8 種のパターンがある。エンベロープ周期は図中の $t_e$の長さだと思えばよい。

$$A_e = R_{11} + R_{12} \times 256$$

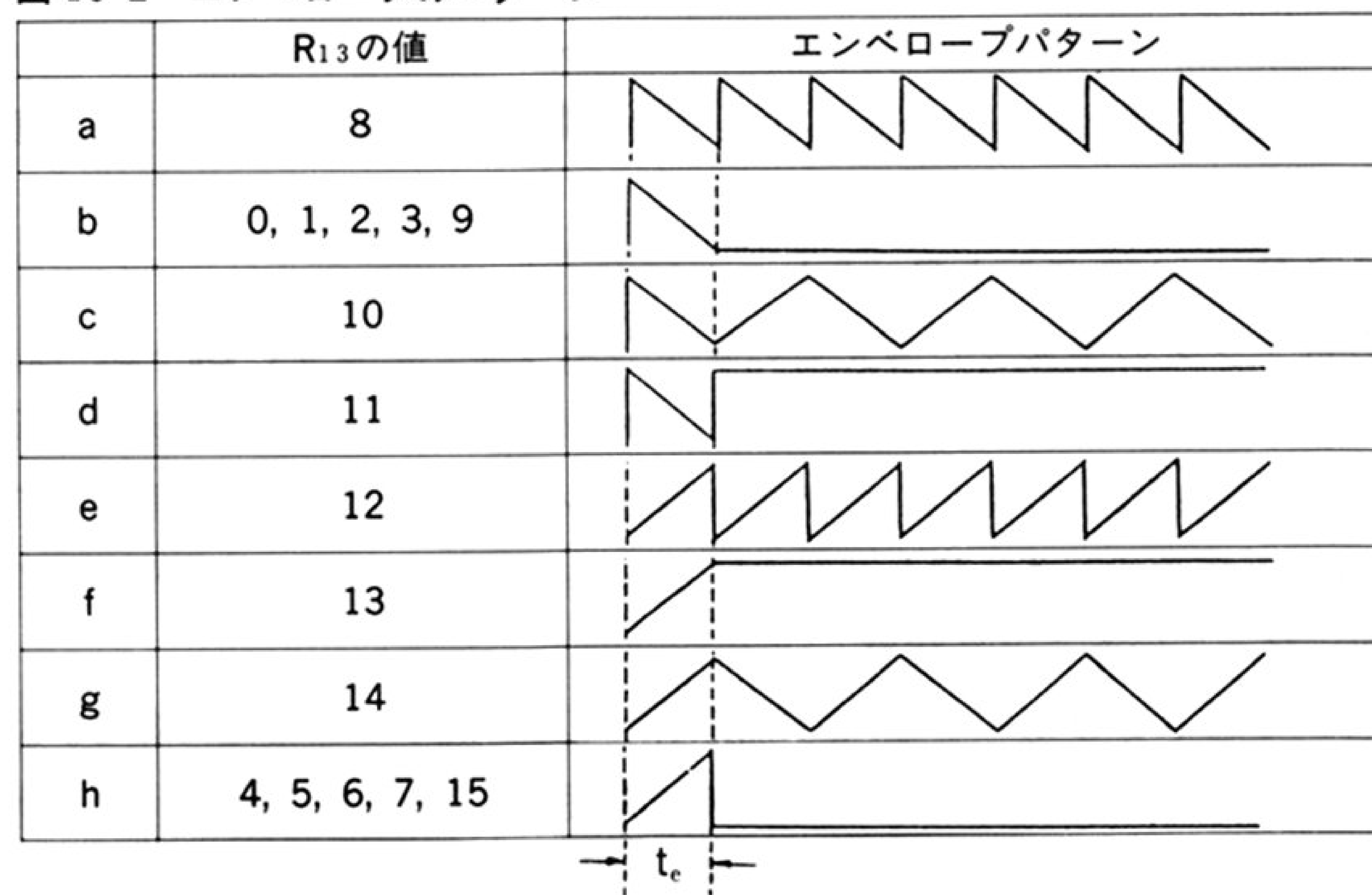
とすると，

$$t_e = \frac{256 \times A_e}{2 \times 10^6}$$

で計算できる。

エンベロープというのは音量を自動的に変えるもので，たとえば「1 秒間だけ音を出して，その後は出さない」などということを，PSG が勝手にやってくれる機能である。図 10-2 のエンベロープパターンを見れば分かるように，音量アップ/ダウンを延々と繰り返すパターンもある。

図 10-2　エンベロープのパターン



| | $R_{13}$の値 | エンベロープパターン |
|---|---|---|
| a | 8 | |
| b | 0, 1, 2, 3, 9 | |
| c | 10 | |
| d | 11 | |
| e | 12 | |
| f | 13 | |
| g | 14 | |
| h | 4, 5, 6, 7, 15 | |

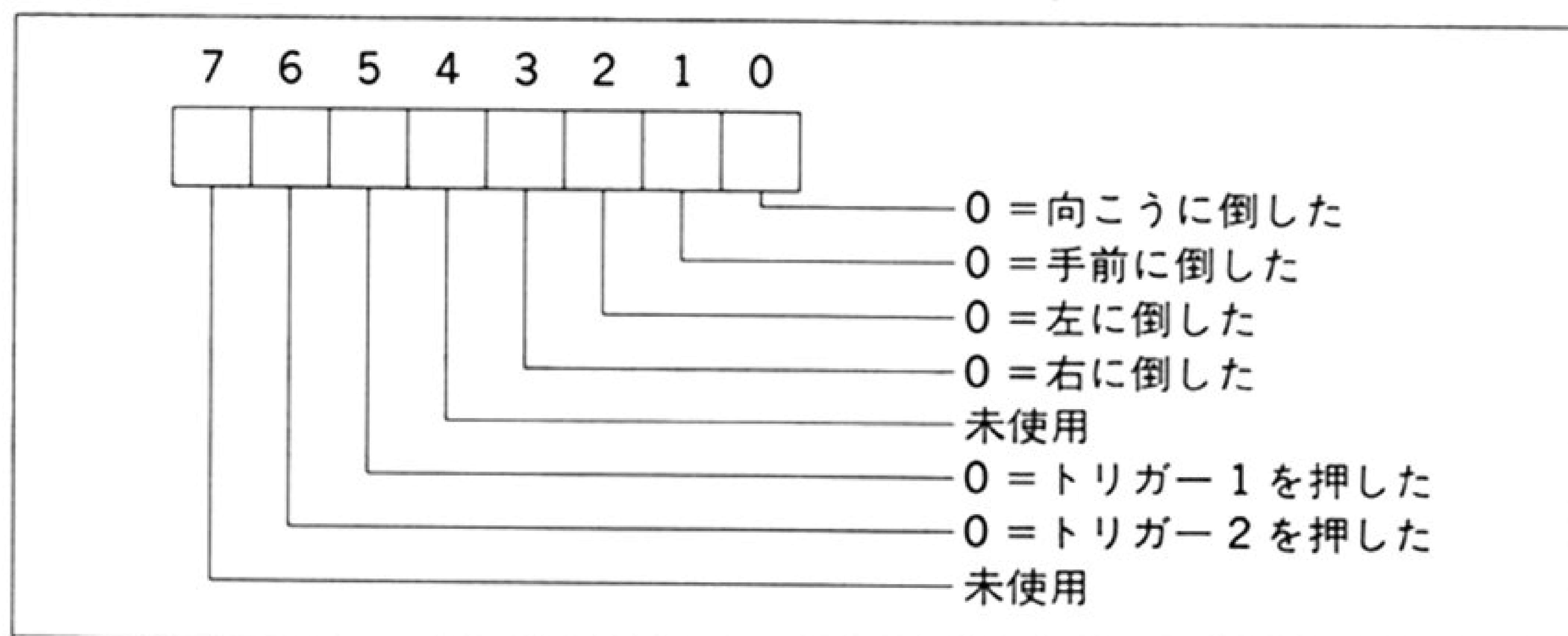
**⑧ $R_{13}$（エンベロープパターン指定）**

図 10-2 に示すエンベロープパターンのうち，どれを使うかを指定するものである。パターン b，h は複数の $R_{13}$の値に対応しているが，「$R_{13}$には 8 ～15 の値だけを設定するものだ」と思っていればよいのだ。

**⑨ $R_{14}$，$R_{15}$（I/O ポート入出力）**

前にも書いたように，X1 ではジョイスティックポートの入出力に使われている。$R_{14}$がジョイスティック 1，$R_{15}$がジョイスティック 2 である。**図 10-3** 参照。

図 10-3　ジョイスティックのデータの意味($R_{14}$，$R_{15}$)




**⑩ 最後に PSG へのアクセス法である。**

まず，レジスタ番号を指定する。つまり，

OUT &H1C00，レジスタ番号（0 ～15)

の後に，

OUT &H1B00，データ

で，データを書き込める。逆にデータを読み出すなら，

～＝INP（&H1B00)

となる。

以上で基礎編の説明を終えたわけである。そして，ここでサンプルプログラムが発動するのだ。

# サンプルなのである

**リスト 10-1**：ドミソ

表 10-2 を使えば一発で分かるはずである。チャンネル A，B，C にそれぞれドミソを割り当て，$R_7$で三つともトーンのみを指定。後は音量を三つとも 15 にしておしまいである。

リスト 10-1　ドミソ

```
100 MUSIC "R0"
110 SOUND 0,&HDD
120 SOUND 1,&H1 :'ト゛
130 SOUND 2,&H7B
```

```
140 SOUND 3,&H1 :'ミ
150 SOUND 4,&H3E
160 SOUND 5,&H1 :'ソ
170 SOUND 7,&B111000
180 SOUND 8,15
190 SOUND 9,15
200 SOUND 10,15
```

さて，このプログラムが実にうっとうしいと思うわけである。そこで**リスト 10-2** である。まず，サブルーチン"QUIET"は PSG を黙らせるサブルーチン，"XSOUND"は PSG のレジスタに指定されたデータを次々と設定するサブルーチンである。"QUIET"を使うには，ただ呼ぶだけでよい。"XSOUND"は少々複雑である。使うには，まず DATA 文を用意する。約束ごとは次の五つである。

**1)** 最初は，レジスタ番号をデータとする。たとえば「1」ならば $R_1$ から設定を始めることになる。

**2)** レジスタに設定したいデータは，0～255 まで，&H～，&B～などもよい。

**3)** 設定したくないレジスタに対応するデータは「S」とする。つまり $R_4$ と $R_6$ だけを 0 にして，$R_5$ をそのままにしておきたければ，

DATA　4，0，S，0，……

とすればよい。

**4)** DATA 文の最後は「EOD」(End Of Data) とする。

**5)** GOSUB "XSOUND" の直前に，

RESTORE（データ文の行番号，もしくはラベル）

を実行しておく。

てなとこである。もっとも，それは X1 ユーザーに限ってのことで，turbo ではサウンド文が拡張されていて，"XSOUND"に相当することがそのまま SOUND 文で実行できるから，きっと，むっとしているはずである。人生とはそんなものである。ところで，リスト 10-2 で行番号がいきなり 1000 に飛んでいることからも分かるように，これらのサブルーチンはほかのリストでも流用される運命にある。心得ておいていただきたい。

**リスト 10-2　PSG 用サブルーチン**

```
100 GOSUB"QUIET"
110 RESTORE 120:GOSUB"XSOUND"
120 DATA 0,&HDD,&H01,&H7B,&H01,&H3E,&H01,S,&B111000,15,15,15,EOD
130 PAUSE 10
140 GOSUB"QUIET"
150 END
160 '
1000 LABEL"QUIET"
1010 SOUND 8,0:SOUND 9,0:SOUND 10,0
1020 RETURN
1030 '
1040 LABEL"XSOUND"
1050 READ C
1060 READ A$:IF A$="EOD"THEN RETURN
1070 IF A$="S" THEN GOTO1090
1080 SOUND C,VAL(A$)
1090 C=C+1:GOTO1060
```

**リスト 10-3** ：銃声

エンベロープパターンはｂを使っている。エンベロープ周期は $R_{11}=0$，$R_{12}=20$ だから，

$$t_e=\frac{256\times(20\times256+0)}{2\times10^6}\fallingdotseq0.66(秒)$$

となる。つまり，最初はボリューム最大で，0.66秒の間に0まで減少するのである。3チャンネルともノイズモードにしてある。130行を見ても分かるように，何かキーを押すと音が出る。押し続けると，機関銃となるわけである。

リスト 10-3　銃声

```
100 GOSUB"QUIET"
110 RESTORE 120:GOSUB"XSOUND"
120 DATA 6,20,&B000111,16,16,16,0,20,9,EOD
130 IF INKEY$(0)="" THEN 130 ELSE 110
140 END
150 '
1000 LABEL"QUIET"
1010 SOUND 8,0:SOUND 9,0:SOUND 10,0
1020 RETURN
1030 '
1040 LABEL"XSOUND"
1050 READ C
1060 READ A$:IF A$="EOD"THEN RETURN
1070 IF A$="S" THEN GOTO1090
1080 SOUND C,VAL(A$)
1090 C=C+1:GOTO1060
```

**リスト 10-4** ：UFO

エンベロープパターンはｇで，3チャンネルともトーン。この手の音を作るのは楽なのである。

リスト 10-4　UFO

```
100 GOSUB"QUIET"
110 RESTORE 120:GOSUB"XSOUND"
120 DATA 0,100,0,102,0,200,0,S,&B111000,16,16,16,10,5,14,EOD
130 PAUSE 50
140 GOSUB"QUIET"
150 END
160 '
1000 LABEL"QUIET"
1010 SOUND 8,0:SOUND 9,0:SOUND 10,0
1020 RETURN
1030 '
1040 LABEL"XSOUND"
1050 READ C
1060 READ A$:IF A$="EOD"THEN RETURN
1070 IF A$="S" THEN GOTO1090
1080 SOUND C,VAL(A$)
1090 C=C+1:GOTO1060
```

**リスト 10-5** ：SLとヘリコプター

RUNすると，最初にSLの音を出す。次にヘリコプターの音である。ヘリコプターの方は，チャンネルBがノイズだけである。そのつもりになって聞くと，そう聞こえるものな

のである。

リスト 10-5 SLとヘリコプター

```
100 GOSUB"QUIET"
110 RESTORE 120:GOSUB"XSOUND"
120 DATA 6,10,&B110111,16,0,0,0,6,12,EOD
130 PAUSE 50
140 RESTORE 150:GOSUB"XSOUND"
150 DATA 0,0,5,0,0,0,0,5,&B100110,16,16,0,0,1,10,EOD
160 PAUSE 50
170 GOSUB"QUIET"
180 END
190 '
1000 LABEL"QUIET"
1010 SOUND 8,0:SOUND 9,0:SOUND 10,0
1020 RETURN
1030 '
1040 LABEL"XSOUND"
1050 READ C
1060 READ A$:IF A$="EOD"THEN RETURN
1070 IF A$="S" THEN GOTO1090
1080 SOUND C,VAL(A$)
1090 C=C+1:GOTO1060
```

**リスト 10-6**：波の音

これはよく見かけるサンプルであるが，**リスト 10-7** のように，ランダムにエンベロープ周期を変えてやると本物に近くなる。時間かせぎの PAUSE 文が 140 行にあるが，別にエンベロープ周期にセットする値と関係あるわけではない。何をしても音がずたずたになるわけではないから，適当でよいのだ。

リスト 10-6 波の音

```
100 GOSUB"QUIET"
110 RESTORE 120:GOSUB"XSOUND"
120 DATA 6,25,&B110111,16,0,0,0,99,14,EOD
130 END
140 '
1000 LABEL"QUIET"
1010 SOUND 8,0:SOUND 9,0:SOUND 10,0
1020 RETURN
1030 '
1040 LABEL"XSOUND"
1050 READ C
1060 READ A$:IF A$="EOD"THEN RETURN
1070 IF A$="S" THEN GOTO1090
1080 SOUND C,VAL(A$)
1090 C=C+1:GOTO1060
```

リスト 10-7 ランダムな波の音

```
100 GOSUB"QUIET"
110 RESTORE 120:GOSUB"XSOUND"
120 DATA 6,25,&B110111,16,0,0,0,99,14,EOD
130 T=40+INT(RND(1)*60):PRINTT
140 SOUND 12,T:PAUSE T:GOTO130
150 '
1000 LABEL"QUIET"
1010 SOUND 8,0:SOUND 9,0:SOUND 10,0
1020 RETURN
```

```
1030 '
1040 LABEL"XSOUND"
1050 READ C
1060 READ A$:IF A$="EOD"THEN RETURN
1070 IF A$="S" THEN GOTO1090
1080 SOUND C,VAL(A$)
1090 C=C+1:GOTO1060
```

以上でサンプルの紹介は終わるが，ここまで見てきて，疑問を持った人もいるはずである。それは何かというならば，エンベロープのパターンはいつ始まるかということである。$R_{13}$(エンベロープパターン)に値をセットしたときからなのか？　それとも，$R_8$～$R_{10}$の$D_5$を1にセットした瞬間からなのか？　ということである。すなわち，エンベロープを発動するトリガー（引金）は何なのかである。

答えは$R_{13}$である。ここに値を書き込まれた瞬間から，エンベロープパターンに従って音が出る。これに対して$R_{11}$，$R_{12}$のエンベロープ周期は単なるカウンタの周期ぐらいの意味しかない。だからリスト10-7のランダムな波の音は，それらしく聞こえたのである。

というところでPSGは終わりである。最近はFM音源が花盛りであるが，PSGも使い方によっては非常に強力なので，きちんと押さえておいて欲しいのであった。

# 第11章

# FM音源




FM音源ナハトムジーク

■第11章
# FM音源ナハトムジーク・・・・・・・

この章ではFM音源についてやってしまうのである。となれば目的はMMLである。で，MMLであるが，turboではNEW turbo BASICにより手軽にFM音源を楽しめるようになった。しかしX1ではそうはいかない。てなわけで，ここでサポートするしだいである。ちなみにOh！MZの連載時にはあちこちにバグが入ったりしたが，それらをすべて直してある(はずである)。

さて，X1のFM音源ボードといえばCZ-8BS1なのであるが，このボードの主役はYM2151,別名OPMと呼ばれるLSIなのである。それに対して他の大概のパソコンに使われているFM音源のLSIはYM2203（OPN）なのである。この両者の違いはというと，
● OPMはFM音源を8チャンネル持っており，それぞれのチャンネルごとに左右（もしくは両方）出力を選択できる。またLFO機能が付いている。
● OPNはFM音源を3チャンネルと，SSG音源を3チャンネル（AY-3-8910と同じ機能)持っている。出力はモノラルのみ。LFO機能は付いていないのでタイマ割り込みを使いソフトウェアでコントロールする必要がある。
ということになる。要するにOPNの方には肝心のFM音源が，

**3チャンネルしか**

付いていないのである。それに対してOPMには，

**8チャンネルも**

付いているのである。これがどういうことかというと，同時に演奏できる楽器の数が3対8なのである。そしてこれは数だけの問題にとどまらず，音の厚みに影響するのである。で，OPMとOPNの使い方（データ）の互換性であるが，基本的に両者には互換性がある。ただしエンベロープのタイミングなどが違い，一部に移植できないものもあるらしい。といってもVIPに付属の200音色を見ても分かるように,OPNから移植できないものがあったとしても，別に不都合があるわけではない。

ちなみに，このFM音源ボードであるが，I/Oアドレスをずらして何枚でもスロットにさせるようにしたら非常にその筋だったと思ったのであるが，残念ながらそうはなっていないようである。

というところで，OPMの具体的な説明に入るのであった。

## OPMである

**図11-1**がOPMのI/Oアドレスである。テンポの制御用にZ80 CTCが付いているが，turboおよびturboZで動かす場合はこのボード上のCTCではなく，本体内蔵のCTC(アドレスは$1FA0_H$～$1FA3_H$)を使うことになっている。気を付けるよーに。

さて，OPM には 256（$-\alpha$）個のレジスタがあるわけだ。そして，これらのレジスタにそれなりの値を書き込むことによって，さまざまな音を出せるのである。この点は PSG（AY-3-8910）と似たよーなものである。そしてレジスタに値を入れる方法であるが，たとえば $21_H$番レジスタに $C3_H$を入れるのであれば，

```
OUT  &H700, &H21
OUT  &H701, &HC3
```

とするのである。まったくの自然体である。

**図 11-1　OPM の I/O アドレス**

| | | |
|---|---|---|
| $0700_H$ | YM2151 アドレスポート | OUT |
| $0701_H$ | YM2151 データポート | IN/OUT |
| $0704_H$ | CTC チャンネル 0 | IN/OUT |
| $0705_H$ | CTC チャンネル 1 | IN/OUT |
| $0706_H$ | CTC チャンネル 2 | IN/OUT |
| $0707_H$ | CTC チャンネル 3 | IN/OUT |

**図 11-2，11-3** がそのレジスタマップである。このマップは基本的に $00_H$番～$1F_H$番（途中で所々で抜けているが）と，$20_H$～$FF_H$の二つの部分に分かれているわけだ。アドレスの見方にちょっと注意が必要だからそのつもりで。

それで，図 11-3 を見ると分かるように，OP2 と OP3 でレジスタ番号が逆転してい

**図 11-2　レジスタマップ（$00_H$～$1F_H$）**

| レジスタ番号 | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|
| $01_H$ | TEST | | | | | | LFO RESET | |
| $08_H$ | | OPマスク OP4 | OPマスク OP3 | OPマスク OP2 | OPマスク OP1 | CH No. | | |
| $0F_H$ | NE | | | NFREQ | | | | |
| $10_H$ | CLK $A_1$ | | | | | | | |
| $11_H$ | | | | | | | CLK $A_2$ | |
| $12_H$ | CLK B | | | | | | | |
| $14_H$ | CMS | | FRESET B | FRESET A | IRQ EN B | IRQ EN A | LOAD B | LOAD A |
| $18_H$ | LFRQ | | | | | | | |
| $19_H$ | F | PMD or AMD | | | | | | |
| $1B_H$ | $CT_2$ | $CT_1$ | | | | | W | |

図 11-3　レジスタマップ($20_H$〜$FF_H$)

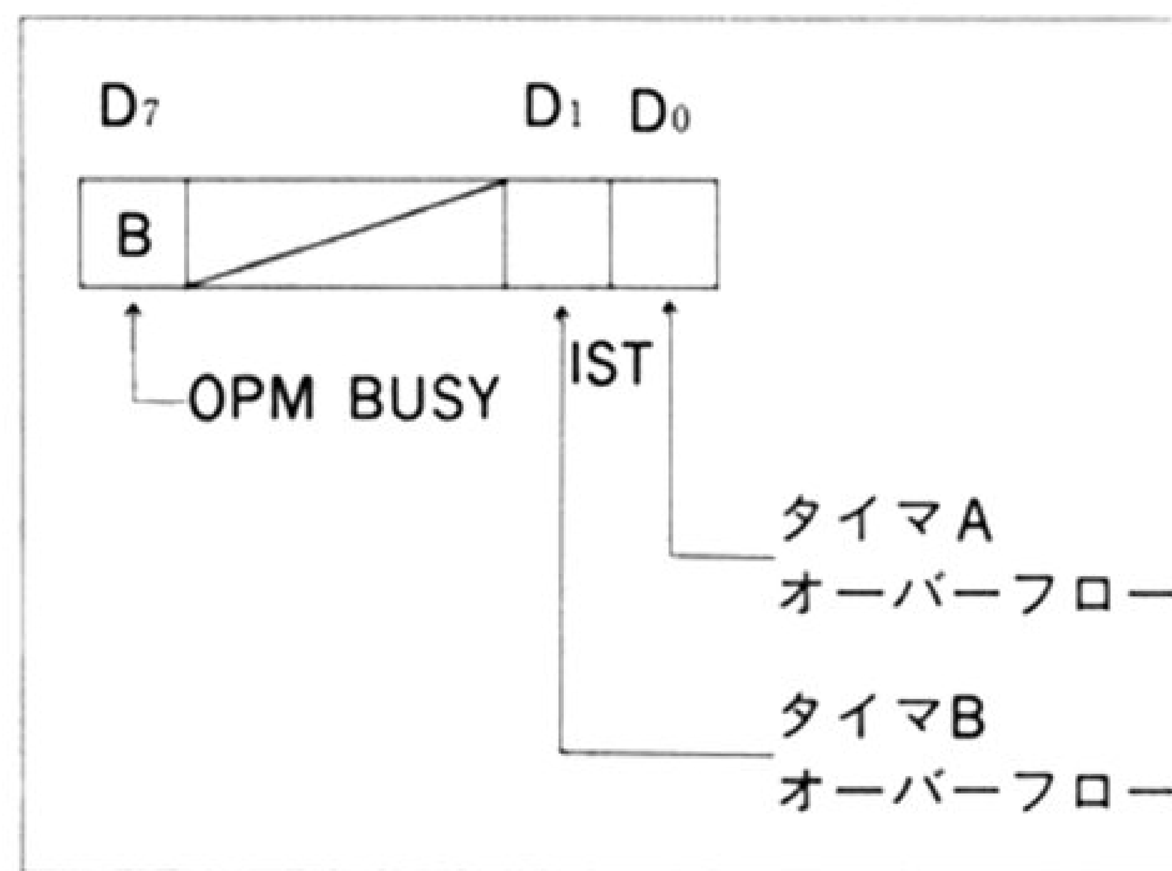
| | CH0 | CH1 | CH2 | CH3 | CH4 | CH5 | CH6 | CH7 | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $20_H$ | $21_H$ | $22_H$ | $23_H$ | $24_H$ | $25_H$ | $26_H$ | $27_H$ | R | L | FEED BACK | | | ALG | | |
| | $28_H$ | $29_H$ | $2A_H$ | $2B_H$ | $2C_H$ | $2D_H$ | $2E_H$ | $2F_H$ | | KEY CODE | | | | | | |
| | $30_H$ | $31_H$ | $32_H$ | $33_H$ | $34_H$ | $35_H$ | $36_H$ | $37_H$ | KEY FRACTION | | | | | | | |
| | $38_H$ | $39_H$ | $3A_H$ | $3B_H$ | $3C_H$ | $3D_H$ | $3E_H$ | $3F_H$ | | PMS | | | | | AMS | |
| OP1 | $40_H$ | $41_H$ | $42_H$ | $43_H$ | $44_H$ | $45_H$ | $46_H$ | $47_H$ | | DT1 | | | MUL | | | |
| OP2 | $50_H$ | $51_H$ | $52_H$ | $53_H$ | $54_H$ | $55_H$ | $56_H$ | $57_H$ | | | | | | | | |
| OP3 | $48_H$ | $49_H$ | $4A_H$ | $4B_H$ | $4C_H$ | $4D_H$ | $4E_H$ | $4F_H$ | | | | | | | | |
| OP4 | $58_H$ | $59_H$ | $5A_H$ | $5B_H$ | $5C_H$ | $5D_H$ | $5E_H$ | $5F_H$ | | | | | | | | |
| OP1 | $60_H$ | $61_H$ | $62_H$ | $63_H$ | $64_H$ | $65_H$ | $66_H$ | $67_H$ | | TL | | | | | | |
| OP2 | $70_H$ | $71_H$ | $72_H$ | $73_H$ | $74_H$ | $75_H$ | $76_H$ | $77_H$ | | | | | | | | |
| OP3 | $68_H$ | $69_H$ | $6A_H$ | $6B_H$ | $6C_H$ | $6D_H$ | $6E_H$ | $6F_H$ | | | | | | | | |
| OP4 | $78_H$ | $79_H$ | $7A_H$ | $7B_H$ | $7C_H$ | $7D_H$ | $7E_H$ | $7F_H$ | | | | | | | | |
| OP1 | $80_H$ | $81_H$ | $82_H$ | $83_H$ | $84_H$ | $85_H$ | $86_H$ | $87_H$ | KS | | | AR | | | | |
| OP2 | $90_H$ | $91_H$ | $92_H$ | $93_H$ | $94_H$ | $95_H$ | $96_H$ | $97_H$ | | | | | | | | |
| OP3 | $88_H$ | $89_H$ | $8A_H$ | $8B_H$ | $8C_H$ | $8D_H$ | $8E_H$ | $8F_H$ | | | | | | | | |
| OP4 | $98_H$ | $99_H$ | $9A_H$ | $9B_H$ | $9C_H$ | $9D_H$ | $9E_H$ | $9F_H$ | | | | | | | | |
| OP1 | $A0_H$ | $A1_H$ | $A2_H$ | $A3_H$ | $A4_H$ | $A5_H$ | $A6_H$ | $A7_H$ | AMS-EN | | | 1DR | | | | |
| OP2 | $B0_H$ | $B1_H$ | $B2_H$ | $B3_H$ | $B4_H$ | $B5_H$ | $B6_H$ | $B7_H$ | | | | | | | | |
| OP3 | $A8_H$ | $A9_H$ | $AA_H$ | $AB_H$ | $AC_H$ | $AD_H$ | $AE_H$ | $AF_H$ | | | | | | | | |
| OP4 | $B8_H$ | $B9_H$ | $BA_H$ | $BB_H$ | $BC_H$ | $BD_H$ | $BE_H$ | $BF_H$ | | | | | | | | |
| OP1 | $C0_H$ | $C1_H$ | $C2_H$ | $C3_H$ | $C4_H$ | $C5_H$ | $C6_H$ | $C7_H$ | DT2 | | | 2DR | | | | |
| OP2 | $D0_H$ | $D1_H$ | $D2_H$ | $D3_H$ | $D4_H$ | $D5_H$ | $D6_H$ | $D7_H$ | | | | | | | | |
| OP3 | $C8_H$ | $C9_H$ | $CA_H$ | $CB_H$ | $CC_H$ | $CD_H$ | $CE_H$ | $CF_H$ | | | | | | | | |
| OP4 | $D8_H$ | $D9_H$ | $DA_H$ | $DB_H$ | $DC_H$ | $DD_H$ | $DE_H$ | $DF_H$ | | | | | | | | |
| OP1 | $E0_H$ | $E1_H$ | $E2_H$ | $E3_H$ | $E4_H$ | $E5_H$ | $E6_H$ | $E7_H$ | 1DL | | | | RR | | | |
| OP2 | $F0_H$ | $E1_H$ | $F2_H$ | $F3_H$ | $F4_H$ | $F5_H$ | $F6_H$ | $F7_H$ | | | | | | | | |
| OP3 | $E8_H$ | $E9_H$ | $EA_H$ | $EB_H$ | $EC_H$ | $ED_H$ | $EE_H$ | $EF_H$ | | | | | | | | |
| OP4 | $F8_H$ | $F9_H$ | $FA_H$ | $FB_H$ | $FC_H$ | $FD_H$ | $FE_H$ | $FF_H$ | | | | | | | | |

る。私は連載時にここで**タコってしまってわんわんわわん，わんわんわわん**だったのである。YAMAHA は一体どういう理由でこんなことをしたのであろうか？　まったく理解に苦しむのである。

ところで，「レジスタに書き込むのは分かったが，それじゃ読み出せるのか？」という疑問が湧くであろう。実はレジスタの値は読み出せないのである。その代わりと言ってはなんだが，OPM では，

INP (&H701)

図 11-4　$701_H$からの入力の内容




とすると，**図 11-4** のようなステータスが得られるのである。これにおいて，ビット 7 は「ただいま私こと OPM は BUSY ですので，データを OUT しないでください」ということなのである。OPM のマニュアルによると直前にデータを書き込んだ後 64 クロックの間 BUSY になるそうである。よって本当はこのビットが 0 になっていることを確認してからデータを送らなければならないのであるが，64 クロックといえばあっという間であるからよほどのことがない限り無視してよいだろう。ビット 1，0 の IST(たぶんインタラプト・ステータスのことであろう）というのはオーバーフロー（割り込み）を起こした（OPM 内蔵の）タイマを示しているのだが，CZ-8BS1 ではこれらの割り込み機能は使えない（使う必要もない)ので説明はしない。どうしても知りたい人は参考文献 6 の YM 2151 のマニュアルを見ていただきたい。また，CZ-8BS1 を持っている人ならば「取扱説明書」も持っているはずであるから，それも参考にしていただきたい。

ではここからねっとりと解説するのであるが，ここの解説だけで OPM がきちんと理解できるとは思わないように。シンセサイザの世界は，また**一味違ったその筋**な世界らしいのである。

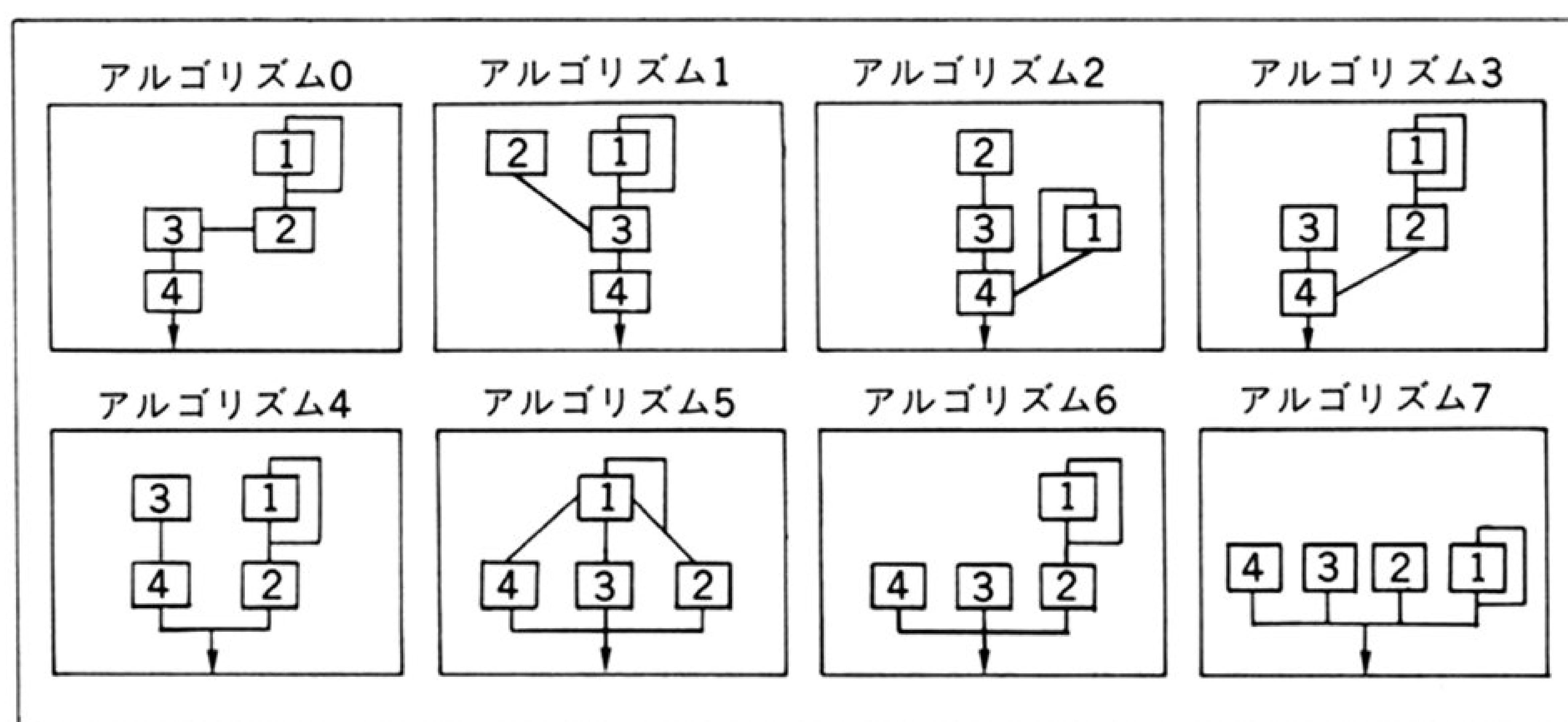
# OPMの基本

OPM には 8 チャンネル（8 ボイス）があるわけなのだが，それぞれのチャンネルはさらに 4 個のオペレータを組み合わせることによってでき上がっている。その四つのオペレータの組み合わせ方には 8 パターンが用意されていて，それが「アルゴリズム」と呼ばれるものである。そのアルゴリズムを**図 11-5** に示す。VIP(CZ-8BS1 に付属のソフト）ではどういうわけか 1～8 となっているのだが，ここでは 0～7 として説明する。ところで，これら八つのアルゴリズムを見ると，1 番(OP1)にだけ自分から出て，また自分に戻る道筋が書かれていることに気付くであろう。これがフィードバック（FEED BACK）というやつである。後で出てくるから覚えておくよーに。

オペレータとは一体いかなるものかというと，基本的にこれは「サイン波発生回路」なのである。ただし単純にサイン波を出すだけのものではなく，あれこれといじることが可能で，それを利用していろいろな音を作れるのである。

さて，シンセサイザのイロハとも言えるのが**図 11-6** の「エンベロープ」である(SSG：

図 11-5　アルゴリズム８態




AY-3-8910 のエンベロープとはちょいと違うので注意)。これは例のアタックレート，ディケイレート，サスティーンレート，サスティーンレベル，リリースレートというやつである。これは出力の時間変化なのである。で，先程も書いたように OPM には 8 チャンネルあり，それぞれにオペレータが 4 個ずつあるのである。つまり全部で 8 × 4 ＝32 個のオペレータがある。そして恐ろしいことに，これら 32 個のオペレータに，それぞれ独立にエンベロープを指定できるのである。さらにはそれぞれのオペレータの周波数を 0.5 倍～25.95 倍の間でずらしたり，さらには微妙にプラス/マイナスしたりも可能なのである。つまりはそのよーにして，微妙に周波数，出力をずらしたサイン波発生回路をあれこれと組み合わせて音を作っていくのである。そしてこれが肝心なのだが，FM 音源において「組み合わせ」というのは「足し合わせる」だけではなく，重関数も可能なのである。すなわち，

$SIN(2\omega t+d_1)+SIN(\omega t+d_2)$

のようなタイプだけではなく，

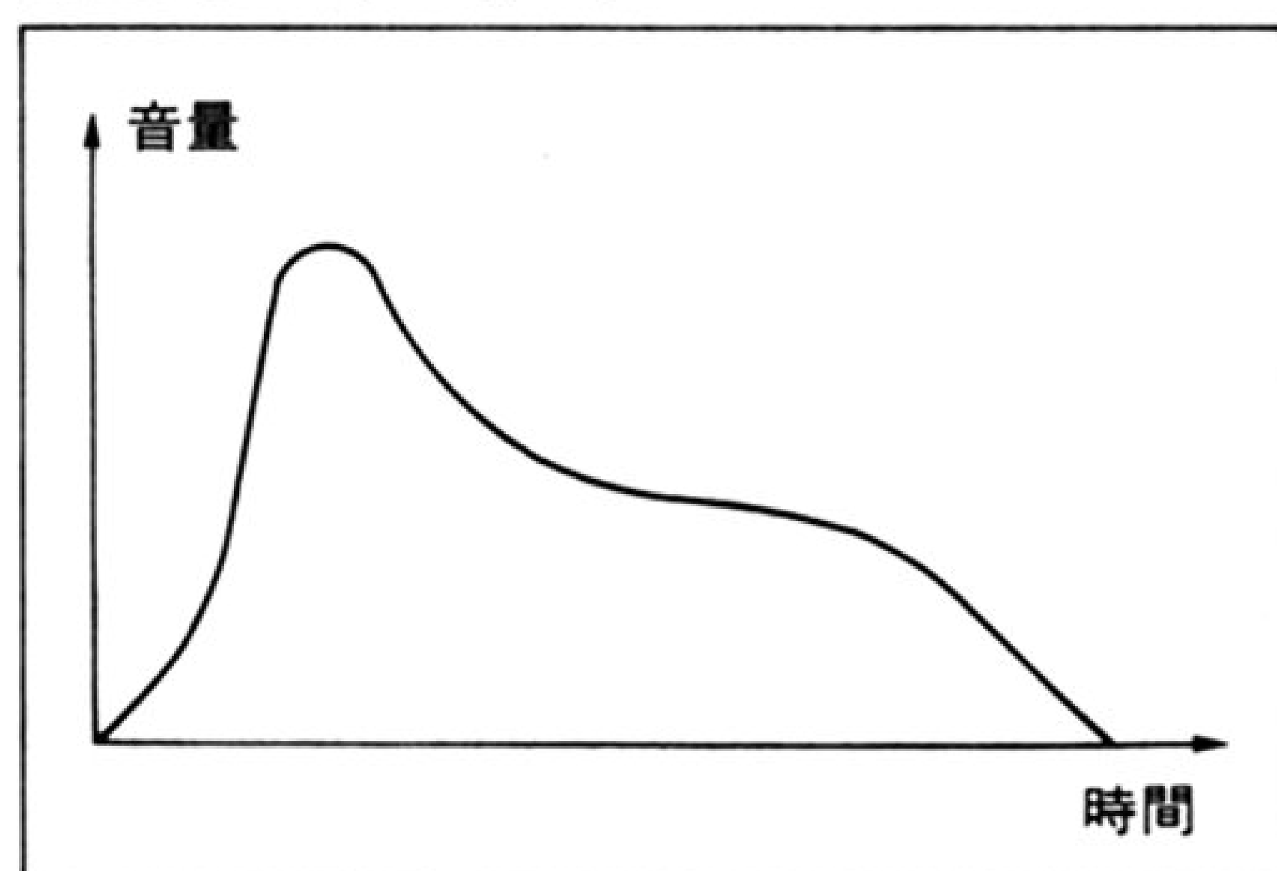
$SIN(\omega SIN(2\omega t+d_1)+d_2)$

などのような組み合わせができるのである。で，オペレータは 4 個もあるわけだ。さらに OP1 は自分の出力を自分に与えたりもできるのである。となると，

$SIN(SIN(SIN(\cdots)+d_1)+d_2)$

というような波形を作ることもできるのである。となれば，「これだけ複雑なことができる

図 11-6　エンベロープ



のだから，きっといろんな音を出すことができるような気がする」であろう。ほ〜ら，あなたはだんだんそんな気になってきた。**ワン，ツウ，スリー**である。

# OPMのレジスタである

というところで，ねっとりとレジスタの解説を始めるのである。

● $01_H$：LFO RESET（TEST）

このレジスタはもともと工場でのテスト用にあるものだが，このレジスタの第1ビットを立ち下げると（「1」→「0」）LFOのリセットが行なわれる。

LFOとはLow Frequency Oscillator＝低周波発振回路のことである。これは音を微妙に震わせるために付いているのだ。音の震わせ方には，

ビブラート：音程（周波数）が上下する

トレモロ：音量（音の大きさ）が上下する

の2とおりがある。LFOはこれらの「震え」の種になるものなのである。LFOには**図11-7**に示すような波形が用意されており，リセットとはこれらの「波形の左端から始めるようにする」ということなのである。

ちなみにSSG（AY-3-8910）の「エンベロープ」はトレモロに相当するものである。

図 11-7 LFO の波形

| W | PMD（ビブラート） | AMD（トレモロ） |
|---|---|---|
| 0（ノコギリ波） | + 0 − | 0 |
| 1（方形波） | + 0 − | 0 |
| 2（三角波） | + 0 − | 0 |
| 3（ノイズ） | + 0 − | 0 |

● $08_H$：KEY ON/OFF

これはそれぞれのチャンネルのそれぞれのオペレータのON/OFFを指定するものである。たとえばこのレジスタに「$1011010_B$」を書き込むと，下3ビットが$010_B$＝2で，その

上の4ビットが$1011_B$だから，チャンネル2のOP4，OP2，OP1がON，OP3がOFFになるのである。たいていの場合はオペレータをすべて使うから，ここにチャンネルナンバーを書けばOFF，それに$+1111000_B=78_H$したものを書けばONということになる。

● $0F_H$：NE/NFREQ

OPMでは第7チャンネルのOP4だけがノイズモードを持っているのである。NE＝1でノイズモードになる。NFREQはSSGのR6（ノイズ周波数）と同じようなものである。5ビット幅で大体の周波数を決めるのである。ノイズに周波数というのも変なのであるが，ま，そんなものなのである。ところでOPMではこのノイズ機能はほとんどお呼びではないのである。現実にVIPはこの機能を使っていないのである。なぜかというと，フィードバックを使うとノイズに近い音が出せてしまうからなのである。よって「チャンネル7のOP4だけ」という制限も加わって，この機能は無視する。

● $10_H$，$11_H$，$12_H$：CLKA，B

これはOPM内蔵のタイマの設定値である。CZ-8BS1では使わないのでいい加減に説明しておく。まずタイマAは10ビットからなる。$A_1$が上位，$A_2$が下位である。その10ビットの値（0〜1023）を$N_A$とすると，タイマAの周期$T_A$は，

$$T_A=\frac{64\times(1024-N_A)}{f_M}$$

となる。$f_M$はOPMに供給されているクロックで，X1では4MHzである。よって$T_A$の最大値は16.38ms，最小値は16$\mu$sである。

タイマBは8ビットで，

$$T_B=\frac{1024\times(256-\mathrm{CLKB})}{f_M}$$

となっている。よって，$256\mu s<T_B<65.536\,ms$である。

● $14_H$：タイマコントロール

LOADがタイマの動作開始，IRQENが割り込みを起こすかどうかのフラグ（CZ-8BS1では使えない），F RESETがオーバーフローフラグ（前述のISTである）のリセットとなっている。どれも「1」で機能する。CSM＝1とするとタイマAがオーバーフローしたときにすべてのオペレータを同時にONにすることができる。ちょっと目には便利そうであるが，割り込み機能が使えないのでほとんど意味がない。ちなみになぜこの割り込み機能が使えないかというと，Z80のモード2の割り込みに対応できないからなのだ。PIOを間に入れればどうにかできるが，それならいっそのこと汎用性の高いCTCを使ってしまえということになる。

なお，オーバーフローが起きればISTの対応するビットは1になっている。

● $18_H$：LFRQ

LFOの周波数を決めるレジスタである。LFRQの値と実際の周波数の関係は**表11-1**のようになっている。この表には規則性があって，

$$f_l\fallingdotseq\frac{f_M\times2^{(\mathrm{LFRQ}/16)}}{4295\times10^6}$$

となっている。

**表 11-1　LFO の周波数**

| LFRQ | 周波数 (Hz) | LFRQ | 周波数 (Hz) | LFRQ | 周波数 (Hz) | LFRQ | 周波数 (Hz) |
|---|---|---|---|---|---|---|---|
| FF | 59.1278 | BF | 3.6955 | 7F | 0.2310 | 3F | 0.0144 |
| FE | 57.2205 | BE | 3.5763 | 7E | 0.2235 | 3E | 0.0140 |
| FD | 55.3131 | BD | 3.4571 | 7D | 0.2161 | 3D | 0.0135 |
| FC | 53.4058 | BC | 3.3379 | 7C | 0.2086 | 3C | 0.0130 |
| FB | 51.4984 | BB | 3.2187 | 7B | 0.2012 | 3B | 0.0126 |
| FA | 49.5911 | BA | 3.0994 | 7A | 0.1937 | 3A | 0.0121 |
| F9 | 47.6837 | B9 | 2.9802 | 79 | 0.1863 | 39 | 0.0116 |
| F8 | 45.7764 | B8 | 2.8610 | 78 | 0.1788 | 38 | 0.0112 |
| F7 | 43.8690 | B7 | 2.7418 | 77 | 0.1714 | 37 | 0.0107 |
| F6 | 41.9617 | B6 | 2.6226 | 76 | 0.1639 | 36 | 0.0102 |
| F5 | 40.0543 | B5 | 2.5034 | 75 | 0.1565 | 35 | 0.0098 |
| F4 | 38.1470 | B4 | 2.3842 | 74 | 0.1490 | 34 | 0.0093 |
| F3 | 36.2396 | B3 | 2.2650 | 73 | 0.1416 | 33 | 0.0088 |
| F2 | 34.3323 | B2 | 2.1458 | 72 | 0.1341 | 32 | 0.0084 |
| F1 | 32.4249 | B1 | 2.0266 | 71 | 0.1267 | 31 | 0.0079 |
| F0 | 30.5176 | B0 | 1.9073 | 70 | 0.1192 | 30 | 0.0075 |
| EF | 29.5639 | AF | 1.8477 | 6F | 0.1155 | 2F | 0.0072 |
| EE | 28.6102 | AE | 1.7881 | 6E | 0.1118 | 2E | 0.0070 |
| ED | 27.6566 | AD | 1.7285 | 6D | 0.1080 | 2D | 0.0068 |
| EC | 26.7029 | AC | 1.6689 | 6C | 0.1043 | 2C | 0.0065 |
| EB | 25.7492 | AB | 1.6093 | 6B | 0.1006 | 2B | 0.0063 |
| EA | 24.7955 | AA | 1.5497 | 6A | 0.0969 | 2A | 0.0061 |
| E9 | 23.8419 | A9 | 1.4901 | 69 | 0.0931 | 29 | 0.0058 |
| E8 | 22.8882 | A8 | 1.4305 | 68 | 0.0894 | 28 | 0.0056 |
| E7 | 21.9345 | A7 | 1.3709 | 67 | 0.0857 | 27 | 0.0054 |
| E6 | 20.9808 | A6 | 1.3113 | 66 | 0.0820 | 26 | 0.0051 |
| E5 | 20.0272 | A5 | 1.2517 | 65 | 0.0782 | 25 | 0.0049 |
| E4 | 19.0735 | A4 | 1.1921 | 64 | 0.0745 | 24 | 0.0047 |
| E3 | 18.1198 | A3 | 1.1325 | 63 | 0.0708 | 23 | 0.0044 |
| E2 | 17.1661 | A2 | 1.0729 | 62 | 0.0671 | 22 | 0.0042 |
| E1 | 16.2125 | A1 | 1.0133 | 61 | 0.0633 | 21 | 0.0040 |
| E0 | 15.2588 | A0 | 0.9537 | 60 | 0.0596 | 20 | 0.0037 |
| DF | 14.7820 | 9F | 0.9239 | 5F | 0.0577 | 1F | 0.0036 |
| DE | 14.3051 | 9E | 0.8941 | 5E | 0.0559 | 1E | 0.0035 |
| DD | 13.8283 | 9D | 0.8643 | 5D | 0.0540 | 1D | 0.0034 |
| DC | 13.3514 | 9C | 0.8345 | 5C | 0.0522 | 1C | 0.0033 |
| DB | 12.8746 | 9B | 0.8047 | 5B | 0.0503 | 1B | 0.0031 |
| DA | 12.3978 | 9A | 0.7749 | 5A | 0.0484 | 1A | 0.0030 |
| D9 | 11.9209 | 99 | 0.7451 | 59 | 0.0466 | 19 | 0.0029 |
| D8 | 11.4441 | 98 | 0.7153 | 58 | 0.0447 | 18 | 0.0028 |
| D7 | 10.9673 | 97 | 0.6855 | 57 | 0.0428 | 17 | 0.0027 |
| D6 | 10.4904 | 96 | 0.6557 | 56 | 0.0410 | 16 | 0.0026 |
| D5 | 10.0136 | 95 | 0.6258 | 55 | 0.0391 | 15 | 0.0024 |
| D4 | 9.5367 | 94 | 0.5960 | 54 | 0.0373 | 14 | 0.0023 |
| D3 | 9.0599 | 93 | 0.5662 | 53 | 0.0354 | 13 | 0.0022 |
| D2 | 8.5831 | 92 | 0.5364 | 52 | 0.0335 | 12 | 0.0021 |
| D1 | 8.1062 | 91 | 0.5066 | 51 | 0.0317 | 11 | 0.0020 |
| D0 | 7.6294 | 90 | 0.4768 | 50 | 0.0298 | 10 | 0.0019 |
| CF | 7.3910 | 8F | 0.4619 | 4F | 0.0289 | 0F | 0.0018 |
| CE | 7.1526 | 8E | 0.4470 | 4E | 0.0279 | 0E | 0.0017 |
| CD | 6.9141 | 8D | 0.4321 | 4D | 0.0270 | 0D | 0.0017 |
| CC | 6.6757 | 8C | 0.4172 | 4C | 0.0261 | 0C | 0.0016 |
| CB | 6.4373 | 8B | 0.4023 | 4B | 0.0251 | 0B | 0.0016 |
| CA | 6.1989 | 8A | 0.3874 | 4A | 0.0242 | 0A | 0.0015 |
| C9 | 5.9605 | 89 | 0.3725 | 49 | 0.0233 | 09 | 0.0015 |
| C8 | 5.7220 | 88 | 0.3576 | 48 | 0.0224 | 08 | 0.0014 |
| C7 | 5.4836 | 87 | 0.3427 | 47 | 0.0214 | 07 | 0.0013 |
| C6 | 5.2452 | 86 | 0.3278 | 46 | 0.0205 | 06 | 0.0013 |
| C5 | 5.0068 | 85 | 0.3129 | 45 | 0.0196 | 05 | 0.0012 |
| C4 | 4.7684 | 84 | 0.2980 | 44 | 0.0186 | 04 | 0.0012 |
| C3 | 4.5300 | 83 | 0.2831 | 43 | 0.0177 | 03 | 0.0011 |
| C2 | 4.2915 | 82 | 0.2682 | 42 | 0.0168 | 02 | 0.0010 |
| C1 | 4.0531 | 81 | 0.2533 | 41 | 0.0158 | 01 | 0.0010 |
| C0 | 3.8147 | 80 | 0.2384 | 40 | 0.0149 | 00 | 0.0009 |

● $19_H$ : F, PMD or AMD

ビット 7 は指定した値が PMD 用であるか AMD 用であるかを区別するためのものである。F＝1 で PMD に，F＝0 で AMD に使われる。想像するに，OPM の内部には PMD 用と AMD 用の**二つ**の 7 ビットの隠れワークエリアがあるのだ。そしてビット 7＝F は，ビット 0～6 の値を PMD のワークエリアの方に送るか，それとも AMD の方に送るかのフラグになっているわけである。

AMD, PMD は 0～127 の値を設定するわけだが，これは早い話が，LFO のかかり具合を決めるものである。ただし，ここだけですべて決まるものではない。詳しくは後程説明する。

ところでどーして素直に $19_H$ と $1A_H$ の二つのレジスタに分けなかったのだろう。不可解である。

● $1B_H$ : $CT_1$, $CT_2$, W

$CT_1$, $CT_2$ は外部をコントロールするためのものである。YM2151 には $CT_1$, $CT_2$ というピンがあって，そこに直行しているのである。つまり 2 ビットだけの汎用出力ポートである。X1 では使えない。

W は LFO の波形を指定するものである。値と波形の関係は図 11-7 である。LFO が PMD に使われるか AMD に使われるかで少し違うことに注意。

ここから先は図 11-3 の方に差しかかってくるわけである。8 個のチャンネルはすべて同じであるからこれから先はチャンネル 0 を対象とする。もしもチャンネル 3 を扱うのならレジスタ番号を＋3 すればよいだけ（以下同様）の話なのである。

● $20_H$ : RL, FB, ALG

ビット 7，6 はそのチャンネルの出力の右左を決めるものである。図 11-3 では左側になっているビット 7 がR（右出力指定）でビット 6 がLであることに注意。FBはOP1のフィードバック（自己変調）の深さを決めるものである。0 ならば自己変調はなしである。資料には**表 11-2** が載っているが，「$\pi/8$」が一体何を表しているのかは，私の知ったことではない。とにかく FB を大きくするとノイズっぽい「ジャ～ン」という音になっていく。ALG は前述のアルゴリズムを指定するものである。

**表 11-2　フィードバックの深さ**

| FB | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| レベル | OFF | $\pi/16$ | $\pi/8$ | $\pi/4$ | $\pi/2$ | $\pi$ | $2\pi$ | $4\pi$ |

● $28_H$ : KEY CODE

● $30_H$ : KEY FRACTION

これは音階を指定するレジスタである。KEY CODE の方はドレミの音階，KEY FRACTION の方は微調整用である（半音＝100 セントとして 1.6 セント刻みで 64 段階＝0～100.8 セントを調整できる）。さて，OPM は実は 3.58 MHz で使うように設計されているのだが，CZ-8BS1 では 4MHz で動かしているのだ。よって音階の指定は本来のデータと違ってしまうのである。それでどうするかというと，KEY FRACTION に 5 を指定

し，後はKEY CODEを半音二つ分下げてやるのである。よって本来の3.58 MHzでは

ド，　レ，　ミ，　ファ，　ソ，　ラ，　シ，　ド　は，

$3E_H$，$41_H$，$44_H$，$45_H$，　$48_H$，$4A_H$，$4D_H$，$4E_H$

に対応するのであるが，CZ-8BS1では，

$3C_H$，$3E_H$，$41_H$，$42_H$，　$45_H$，$48_H$，$4A_H$，$4C_H$

となるのである。ドが「？$C_H$」になったのであるから分かりやすくなったとも思えるがよう分からん。なお，KEY CODEに$10_H$を加えると1オクターブ上の音になるのである。**表11-3**を参照のこと。

表11-3　4MHz動作時のキーコード(KEY FRACTION＝5とすること)

| KEY CODE | $+0C_H$ | $+0D_H$ | $+0E_H$ | $+0F_H$ | $+10_H$ | $+11_H$ | $+12_H$ | $+13_H$ | $+14_H$ | $+15_H$ | $+16_H$ | $+17_H$ | $+18_H$ | $+19_H$ | $+1A_H$ | $+1B_H$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 音　程 | C | C# | D | (D#) | D# | E | F | (F#) | F# | G | G# | (A) | A | A# | B | (+C) |

● $38_H$：PMS，AMS

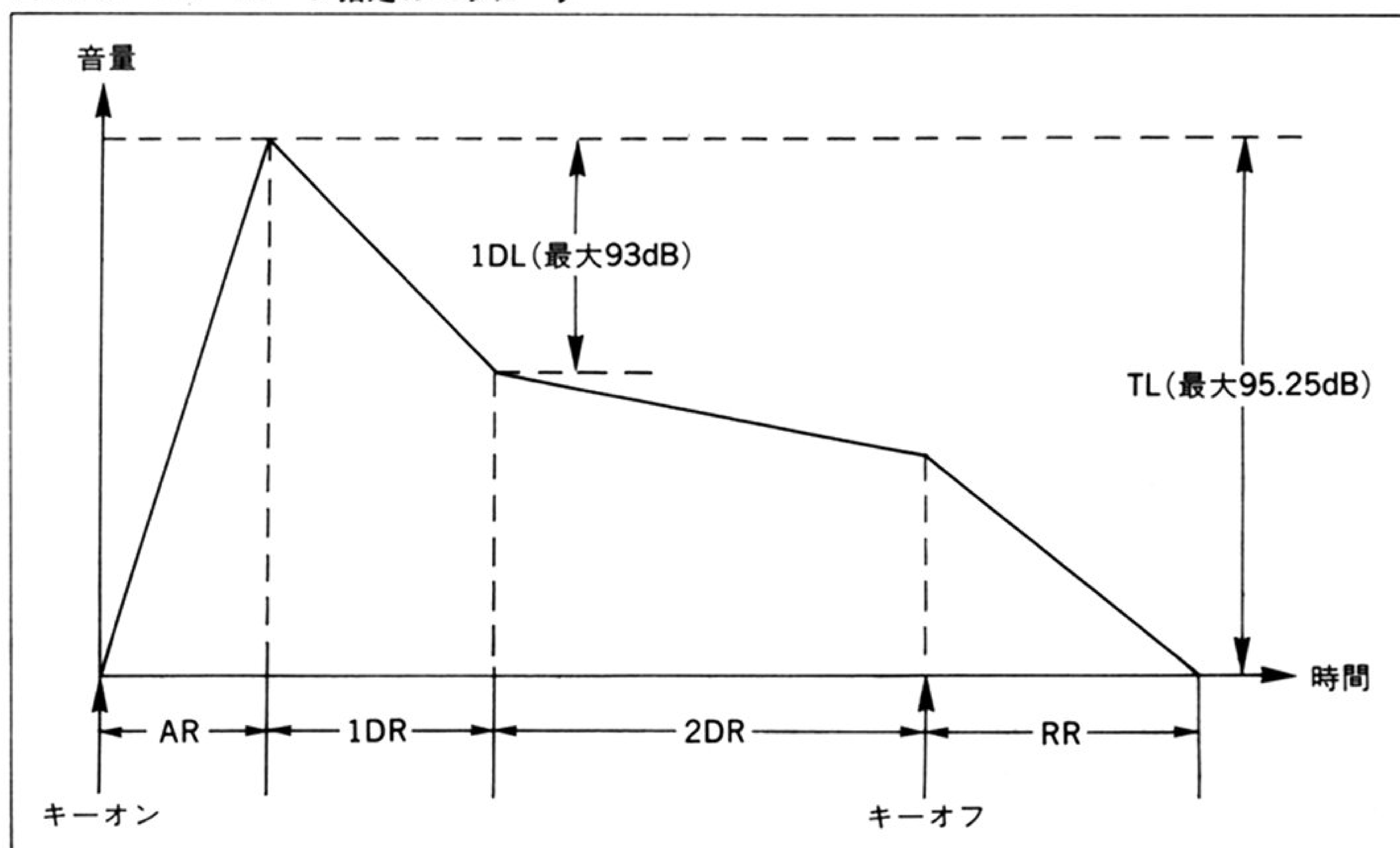
これは$19_H$のPMD or AMDと関係あるところである。後でまとめて解説する。

● $40_H$以降はレジスタ番号順ではなく，機能ごとにまとめて説明する。

① **AR，1DR，2DR，1DL，RR，TL，KS**

これらはエンベロープに関係するパラメータである。**図11-8**に示したような機能を持っている。図からは逆のような感じがするかもしれないが，AR(アタックレート)，1DR(ファーストディケイレート)，2DR（セカンドディケイレート)，RR（リリースレート）は大きければ大きいほどエンベロープの変化が速くなるのである。1DL(ファーストディケイレベル)は少し違って，1DR→2DRの切り換え点を示すものである。ARで最大に達した出

図11-8　エンベロープ指定のパラメータ



力が1DRで指定された速度で1DL分減ったところで減り方が2DRに従うように切り換えられるのである。つまり，

1) AR：キーオンになったら，とにかくこの速さで最大の出力に達しなさい（キーオン，キーオフはレジスタ$08_H$で指定するんでしたねっ)。

2) 1DR：ARで最大に達した後はこの速さで減っていきなさい。ただし1DLで指定された分だけですよ。

3) 2DR：1DL分減りましたね。では今からこの2DRで指定する速さで減っていきなさい。ただしキーオフになるまでです。

4) RR：キーオフになった後はRRで指定した速さで減りなさい。はいよくできました。

となっているのである。そして最後のTL（トータルレベル）は要するに音量なのである。TLは0で最大，127で最小の出力となる。逆であるから注意のこと。

で，問題は「速さ」は具体的にどうなっているのかである。それは**表11-4，11-5**なのであるが，これにはKS(キースケーリング)が絡んでくるのである。キースケーリングとはどういうものかというと，

**音の高さに応じてエンベロープの変化も速くする**（そういうモードも付けておこう）

表11-4　KS, KC′→$R_{KS}$

| KC′＼KS | 0 | 1 | 2 | 3 | KC′＼KS | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 16 | 2 | 4 | 8 | 16 |
| 1 | 0 | 0 | 0 | 1 | 17 | 2 | 4 | 8 | 17 |
| 2 | 0 | 0 | 1 | 2 | 18 | 2 | 4 | 9 | 18 |
| 3 | 0 | 0 | 1 | 3 | 19 | 2 | 4 | 9 | 19 |
| 4 | 0 | 1 | 2 | 4 | 20 | 2 | 5 | 10 | 20 |
| 5 | 0 | 1 | 2 | 5 | 21 | 2 | 5 | 10 | 21 |
| 6 | 0 | 1 | 3 | 6 | 22 | 2 | 5 | 11 | 22 |
| 7 | 0 | 1 | 3 | 7 | 23 | 2 | 5 | 11 | 23 |
| 8 | 1 | 2 | 4 | 8 | 24 | 3 | 6 | 12 | 24 |
| 9 | 1 | 2 | 4 | 9 | 25 | 3 | 6 | 12 | 25 |
| 10 | 1 | 2 | 5 | 10 | 26 | 3 | 6 | 13 | 26 |
| 11 | 1 | 2 | 5 | 11 | 27 | 3 | 6 | 13 | 27 |
| 12 | 1 | 3 | 6 | 12 | 28 | 3 | 7 | 14 | 28 |
| 13 | 1 | 3 | 6 | 13 | 29 | 3 | 7 | 14 | 29 |
| 14 | 1 | 3 | 7 | 14 | 30 | 3 | 7 | 15 | 30 |
| 15 | 1 | 3 | 7 | 15 | 31 | 3 | 7 | 15 | 31 |

**表11-5　出力変化時間表**

**A　10%↔90%**

| ATTACK TIME | | DECAY TIME | |
|---|---|---|---|
| RATE | ms(10%→90%) | RATE | ms(90%→10%) |
| 63 | 0.00 | 63 | 1.22 |
| 62 | 0.24 | 62 | 1.22 |
| 61 | 0.24 | 61 | 1.22 |
| 60 | 0.24 | 60 | 1.22 |
| 59 | 0.30 | 59 | 1.39 |

**B　0%↔100%**

| ATTACK TIME | | DECAY TIME | |
|---|---|---|---|
| RATE | ms(0%→100%) | RATE | ms(100%→0%) |
| 63 | 0.00 | 63 | 6.02 |
| 62 | 0.47 | 62 | 6.02 |
| 61 | 0.47 | 61 | 6.02 |
| 60 | 0.47 | 60 | 6.02 |
| 59 | 0.57 | 59 | 8.03 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 0.36 | 58 | 1.62 | 58 | 0.67 | 58 | 8.03 |
| 57 | 0.42 | 57 | 1.95 | 57 | 0.81 | 57 | 9.63 |
| 56 | 0.59 | 56 | 2.43 | 56 | 1.00 | 56 | 12.04 |
| 55 | 0.55 | 55 | 2.78 | 55 | 1.09 | 55 | 13.77 |
| 54 | 0.65 | 54 | 3.26 | 54 | 1.27 | 54 | 16.06 |
| 53 | 0.78 | 53 | 3.89 | 53 | 1.53 | 53 | 19.27 |
| 52 | 0.98 | 52 | 4.87 | 52 | 1.91 | 52 | 24.08 |
| 51 | 1.12 | 51 | 5.57 | 51 | 1.99 | 51 | 27.52 |
| 50 | 1.31 | 50 | 6.49 | 50 | 2.33 | 50 | 32.11 |
| 49 | 1.57 | 49 | 7.79 | 49 | 2.78 | 49 | 38.53 |
| 48 | 1.96 | 48 | 9.74 | 48 | 3.48 | 48 | 48.16 |
| 47 | 2.24 | 47 | 11.12 | 47 | 3.98 | 47 | 55.04 |
| 46 | 2.61 | 46 | 12.99 | 46 | 4.65 | 46 | 64.22 |
| 45 | 3.13 | 45 | 15.58 | 45 | 5.58 | 45 | 77.06 |
| 44 | 3.91 | 44 | 19.48 | 44 | 6.97 | 44 | 96.33 |
| 43 | 4.48 | 43 | 22.26 | 43 | 7.97 | 43 | 110.09 |
| 42 | 5.22 | 42 | 25.96 | 42 | 9.29 | 42 | 128.43 |
| 41 | 6.27 | 41 | 31.16 | 41 | 11.15 | 41 | 154.12 |
| 40 | 7.87 | 40 | 38.95 | 40 | 13.94 | 40 | 192.65 |
| 39 | 8.95 | 39 | 44.52 | 39 | 15.93 | 39 | 220.17 |
| 38 | 10.44 | 38 | 52.83 | 38 | 18.58 | 38 | 212.12 |
| 37 | 12.52 | 37 | 62.32 | 37 | 22.30 | 37 | 308.25 |
| 36 | 15.65 | 36 | 77.90 | 36 | 27.88 | 36 | 385.31 |
| 35 | 17.89 | 35 | 89.03 | 35 | 31.86 | 35 | 440.35 |
| 34 | 20.87 | 34 | 103.86 | 34 | 41.53 | 34 | 513.74 |
| 33 | 25.05 | 33 | 124.64 | 33 | 44.60 | 33 | 616.48 |
| 32 | 31.32 | 32 | 155.80 | 32 | 55.75 | 32 | 770.60 |
| 31 | 35.78 | 31 | 177.43 | 31 | 63.72 | 31 | 835.95 |
| 30 | 41.75 | 30 | 207.74 | 30 | 74.34 | 30 | 1027.48 |
| 29 | 50.10 | 29 | 249.28 | 29 | 89.20 | 29 | 1232.97 |
| 28 | 62.62 | 28 | 311.60 | 28 | 111.51 | 28 | 1541.22 |
| 27 | 71.57 | 27 | 356.12 | 27 | 127.43 | 27 | 1761.39 |
| 26 | 83.50 | 26 | 415.47 | 26 | 148.68 | 26 | 2296.04 |
| 25 | 100.20 | 25 | 498.57 | 25 | 178.41 | 25 | 2465.94 |
| 24 | 125.26 | 24 | 623.21 | 24 | 223.01 | 24 | 3082.42 |
| 23 | 143.15 | 23 | 712.23 | 23 | 254.87 | 23 | 3522.77 |
| 22 | 167.01 | 22 | 830.94 | 22 | 297.35 | 22 | 4109.90 |
| 21 | 200.40 | 21 | 997.13 | 21 | 356.82 | 21 | 4931.89 |
| 20 | 250.50 | 20 | 1246.41 | 20 | 446.02 | 20 | 6164.86 |
| 19 | 286.29 | 19 | 1424.47 | 19 | 509.74 | 19 | 7045.55 |
| 18 | 334.01 | 18 | 1661.88 | 18 | 594.69 | 18 | 8228.76 |
| 17 | 400.81 | 17 | 1994.26 | 17 | 713.63 | 17 | 9863.77 |
| 16 | 501.01 | 16 | 2492.83 | 16 | 892.04 | 16 | 12329.71 |
| 15 | 572.59 | 15 | 2848.95 | 15 | 1019.48 | 15 | 14091.09 |
| 14 | 668.01 | 14 | 3323.77 | 14 | 1189.38 | 14 | 16439.61 |
| 13 | 801.62 | 13 | 3988.52 | 13 | 1427.27 | 13 | 19727.54 |
| 12 | 1002.02 | 12 | 4985.65 | 12 | 1784.08 | 12 | 24659.42 |
| 11 | 1145.16 | 11 | 5697.88 | 11 | 2038.95 | 11 | 28182.20 |
| 10 | 1336.02 | 10 | 6647.53 | 10 | 2378.78 | 10 | 32879.23 |
| 9 | 1603.23 | 9 | 7977.05 | 9 | 2854.53 | 9 | 39455.07 |
| 8 | 2004.04 | 8 | 9971.30 | 8 | 3568.16 | 8 | 49318.84 |
| 7 | 2290.32 | 7 | 11395.78 | 7 | 4077.90 | 7 | 56364.40 |
| 6 | 2672.05 | 6 | 13295.07 | 6 | 4757.55 | 6 | 65758.46 |
| 5 | 3206.45 | 5 | 15954.08 | 5 | 5709.06 | 5 | 78910.15 |
| 4 | 4008.07 | 4 | 19942.60 | 4 | 7136.33 | 4 | 98637.69 |
| 3 | 無限大 | 3 | 無限大 | 3 | 無限大 | 3 | 無限大 |
| 2 | 無限大 | 2 | 無限大 | 2 | 無限大 | 2 | 無限大 |
| 1 | 無限大 | 1 | 無限大 | 1 | 無限大 | 1 | 無限大 |
| 0 | 無限大 | 0 | 無限大 | 0 | 無限大 | 0 | 無限大 |

というものである。保証の限りではないがそうした方がより自然な音色になるそうなのである。ではキースケーリングを説明する。

まずはRを求める。AR，1DR，2DRの場合ならばそれらの値（レジスタに書き込んだ値）がRそのものであるが，そうではなくて，もしもRRの場合ならば，

R＝2×RR＋1

とするのである。

次に表11-4を使って$R_{KS}$を求める。横のカギがKS(0～3)，縦のカギがKEY CODEの上位5ビット＝KC′(0～31)である。たとえばKS＝2，KEY CODE＝$2E_H$＝$0101110_B$とすると，KC′＝$01011_B$＝11となるので表より$R_{KS}$は5となる。

Rと$R_{KS}$が求められたならば，

RATE＝MAX(2×R＋$R_{KS}$，63)

とする。つまり，RATEの上限を63に制限するのである。このRATEを使い，表11-5 A，11-5 Bから時間を計算することになる。表11-5 Aの方は出力が10％↔90％の変化をする時間，表11-5 Bの方は0％↔100％の変化をする時間となっている。で，両方の表とも左側はアタックレート（AR）によって増加する時間，右側はディケイレート（1DR，2DR，RR）によって減少する時間である。単位はすべてms(1/1000秒）となっている。

あと説明が必要なのが1DLであるが，これは，

減衰量＝1DL×3(dB)

となっている。ただし，1DLが最大の15のときはさらに＋48 dBすることになっている(つまり最大減衰量は15×3＋48＝93 dB)。dBとは減衰量を表す単位で，デシベルと読む。ベルとは電話を発明したグラハム・ベルに由来した単位である。TLの方は，

減衰量＝TL×0.75(dB)

となっており，これの最大減衰量は95.25 dBである。大体の目安となるであろう。

**② DT1，DT2，MUL**

これらKEY CODEとKEY FRACTIONで決めた周波数を微調節するパラメータである。KEY CODEとKEY FRACTIONは各チャンネルに一つだが，DT1，DT2，MULは各オペレータごとに設定が可能である。すなわち一つのチャンネルでオペレータが別々の周波数で動作するのだ。まずは**表11-6**である。横のカギがMUL（0～15)，縦のカギがDT2（0～3）である。この表から得られた値をFr（FREQUENCY RATIO）とする。

表11-6 DT2，MUL→Fr表

| DT2 \ MUL | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0→1倍 | 0.50 | 1.00 | 2.00 | 3.00 | 4.00 | 5.00 | 6.00 | 7.00 | 8.00 | 9.00 | 10.00 | 11.00 | 12.00 | 13.00 | 14.00 | 15.00 |
| 1→ $\sqrt{2}$倍 | 0.71 | 1.41 | 2.82 | 4.24 | 5.65 | 7.07 | 8.46 | 9.89 | 11.30 | 12.72 | 14.10 | 15.55 | 16.96 | 18.37 | 19.78 | 21.20 |
| 2→ $\sqrt{2.5}$倍 | 0.78 | 1.57 | 3.14 | 4.71 | 6.28 | 7.85 | 9.42 | 10.99 | 12.56 | 14.13 | 15.70 | 17.27 | 18.84 | 20.41 | 21.98 | 23.55 |
| 3→ $\sqrt{3}$倍 | 0.87 | 1.73 | 3.46 | 5.19 | 6.92 | 8.65 | 10.38 | 12.11 | 13.84 | 15.57 | 17.30 | 19.03 | 20.76 | 22.49 | 24.22 | 25.95 |

表 11-7　KC′，DT1→Fd 表

| DT1 / KC′ | セント(半音＝100 セント) | | | | Hz | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | ±1 | ±2 | ±3 | 0 | ±1 | ±2 | ±3 |
| 0 | 0.000 | 0.000 | 5.025 | 10.036 | 0.000 | 0.000 | 0.053 | 0.107 |
| 1 | 0.000 | 0.000 | 4.228 | 8.445 | 0.000 | 0.000 | 0.053 | 0.107 |
| 2 | 0.000 | 0.000 | 3.559 | 7.110 | 0.000 | 0.000 | 0.053 | 0.107 |
| 3 | 0.000 | 0.000 | 2.993 | 5.980 | 0.000 | 0.000 | 0.053 | 0.107 |
| 4 | 0.000 | 2.515 | 5.025 | 5.025 | 0.000 | 0.053 | 0.107 | 0.107 |
| 5 | 0.000 | 2.115 | 4.228 | 6.338 | 0.000 | 0.053 | 0.107 | 0.160 |
| 6 | 0.000 | 1.778 | 3.555 | 5.330 | 0.000 | 0.053 | 0.107 | 0.160 |
| 7 | 0.000 | 1.496 | 2.990 | 4.483 | 0.000 | 0.053 | 0.107 | 0.160 |
| 8 | 0.000 | 1.258 | 2.515 | 5.025 | 0.000 | 0.053 | 0.107 | 0.213 |
| 9 | 0.000 | 1.057 | 3.170 | 4.225 | 0.000 | 0.053 | 0.160 | 0.213 |
| 10 | 0.000 | 0.889 | 2.667 | 3.555 | 0.000 | 0.053 | 0.160 | 0.213 |
| 11 | 0.000 | 0.748 | 2.242 | 3.735 | 0.000 | 0.053 | 0.160 | 0.267 |
| 12 | 0.000 | 1.258 | 2.515 | 3.143 | 0.000 | 0.107 | 0.213 | 0.267 |
| 13 | 0.000 | 1.057 | 2.114 | 3.170 | 0.000 | 0.107 | 0.213 | 0.320 |
| 14 | 0.000 | 0.889 | 1.778 | 2.667 | 0.000 | 0.107 | 0.213 | 0.320 |
| 15 | 0.000 | 0.748 | 1.869 | 2.615 | 0.000 | 0.107 | 0.267 | 0.373 |
| 16 | 0.000 | 0.629 | 1.572 | 2.515 | 0.000 | 0.107 | 0.267 | 0.427 |
| 17 | 0.000 | 0.793 | 1.566 | 2.114 | 0.000 | 0.160 | 0.320 | 0.427 |
| 18 | 0.000 | 0.667 | 1.334 | 2.001 | 0.000 | 0.160 | 0.320 | 0.480 |
| 19 | 0.000 | 0.561 | 1.308 | 1.869 | 0.000 | 0.160 | 0.373 | 0.533 |
| 20 | 0.000 | 0.629 | 1.258 | 1.729 | 0.000 | 0.213 | 0.427 | 0.587 |
| 21 | 0.000 | 0.529 | 1.057 | 1.566 | 0.000 | 0.213 | 0.427 | 0.640 |
| 22 | 0.000 | 0.445 | 1.001 | 1.445 | 0.000 | 0.213 | 0.480 | 0.693 |
| 23 | 0.000 | 0.467 | 0.935 | 1.308 | 0.000 | 0.267 | 0.533 | 0.747 |
| 24 | 0.000 | 0.393 | 0.865 | 1.258 | 0.000 | 0.267 | 0.587 | 0.853 |
| 25 | 0.000 | 0.397 | 0.793 | 1.123 | 0.000 | 0.320 | 0.640 | 0.907 |
| 26 | 0.000 | 0.334 | 0.723 | 1.056 | 0.000 | 0.320 | 0.693 | 1.013 |
| 27 | 0.000 | 0.327 | 0.654 | 0.935 | 0.000 | 0.373 | 0.747 | 1.067 |
| 28 | 0.000 | 0.315 | 0.629 | 0.865 | 0.000 | 0.427 | 0.853 | 1.173 |
| 29 | 0.000 | 0.315 | 0.629 | 0.865 | 0.000 | 0.427 | 0.853 | 1.173 |
| 30 | 0.000 | 0.315 | 0.629 | 0.865 | 0.000 | 0.727 | 0.853 | 1.173 |
| 31 | 0.000 | 0.315 | 0.629 | 0.865 | 0.000 | 0.427 | 0.853 | 1.173 |

次に DT1（－3～＋3）と，先程出てきた KEY CODE の上位 5 ビット分の値，KC′を使って**表 11-7** から Fd を得る。DT1 はレジスタ $40_H$～$5F_H$の第 6，5，4 ビットであるが，第 6 ビットを符号ビットと解釈するのである。そして Fd と DT1 の符号は同じにすること。つまり KC′＝11，DT1＝－2 であったなら，Fd＝－0.160(Hz)＝－2.242(セント) となる。

以上のように Fr と Fd が求められたなら，KEY CODE と KEY FRACTION（＋LFO による震え）で決まっていた周波数 F に対して，

$$F' = F \times Fr + Fd$$

としてこの F′を各オペレータの周波数とするのである。繰り返すが F′は各オペレータごとに変えることができるのである。

### ③ PMD，AMD，PMS，AMS，AMD-EN

いよいよ最後である。これは**図 11-9** を見ていただくと一目瞭然であろう。そして，PMD or AMD が最大（つまり 127）のときの PMS のかかり具合は**表 11-8** である。PMS＝7 の

図 11-9 パラメータの流れ

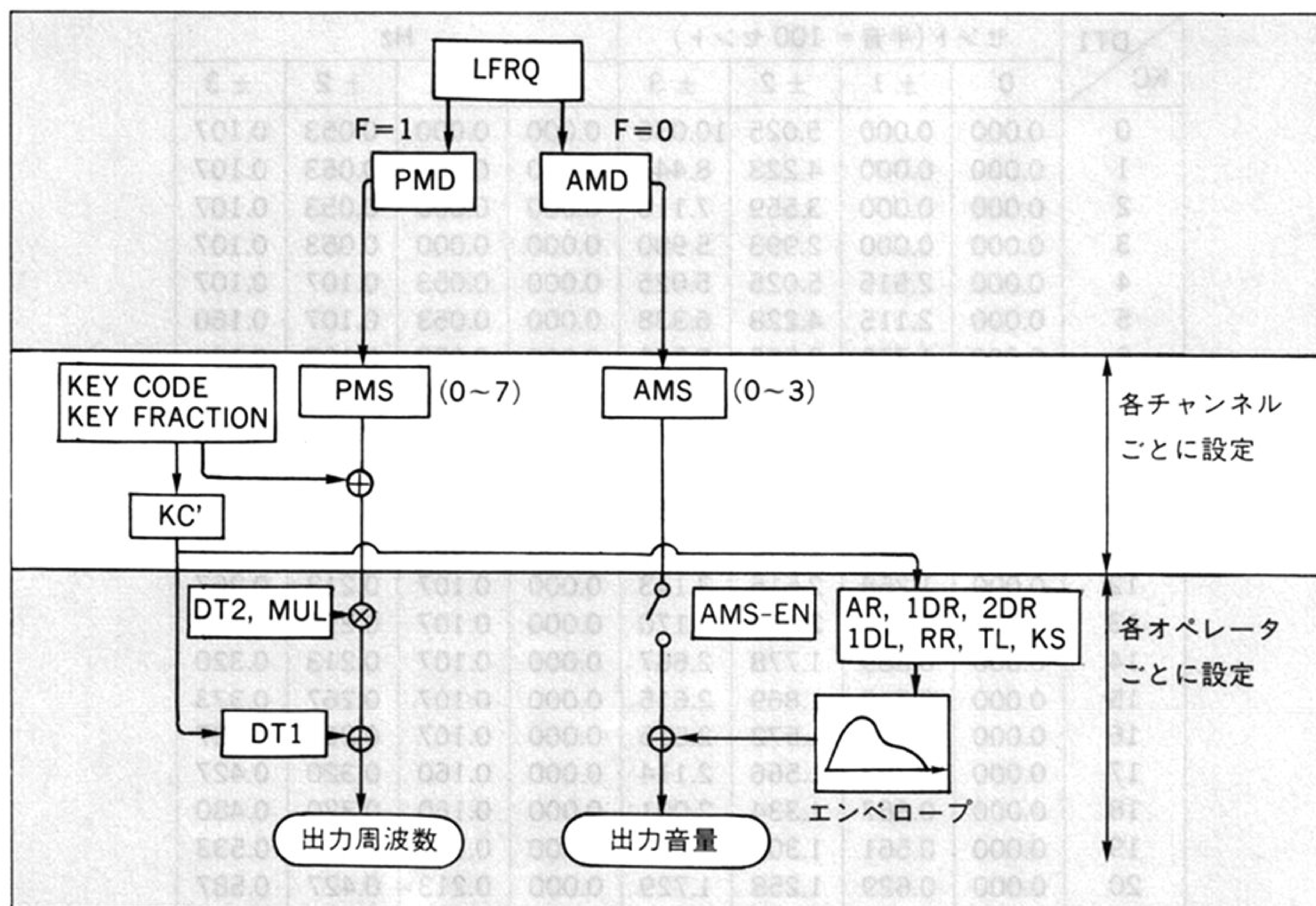



表 11-8 PMS による変調度(PMD=127 のとき)

| PMS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 最大変調度(セント)(半音=100 セント) | 0 | ±5 | ±10 | ±20 | ±50 | ±100 | ±400 | ±700 |

ときは 700 セントであるから，つまりは半音 7 個分なのである。AMD の方は，減衰量は AMD=0 → 0dB, 1 → 23.90625dB, 2 → 47.8125dB, 3 → 95.625dB となっている。TL による最大減衰量が 95.25dB であるから，要するに AMD を最大限にかけると谷底で出力は 0 になってしまうのである。

というところで OPM のレジスタの説明は終わりである。これだけではわけが分からないかもしれないので，ちょいとサンプルを載せておく。**リスト 11-1** が VIP の NEWTONE から持ってきたピアノの音色データである。**リスト 11-2** の 231～232 行を挿入して，480 行以降を書き換えるとドミソの和音となる。なおデータ文中の数値は 2 桁ならば 16 進数，1 桁もしくは 3 桁ならば 10 進数，それ以外ならば 2 進数と解釈されるようになっている。データ分を適当に書き換えて OPM を**いじりまわして**いただきたい。

リスト 11-1 ピアノ

```
100 READ R$:R1$=LEFT$(R$,1):IF R1$="!" THEN END
110 IF R1$="P" THEN PAUSE VAL(MID$(R$,2,255)):GOTO 100
120 R=VAL("&H"+R$)
130 READ D$:DL=INSTR(D$," ")
140 IF DL=0 THEN DL=LEN(D$)+1
```

```
150 ON DL GOTO 170,170,180,170
160 E$="&B":GOTO 190
170 E$="":GOTO 190
180 E$="&H"
190 D=VAL(E$+D$):'IF D<>0 THEN PRINTHEX$(R),D
200 GOSUB"SFM":GOTO 100
210 '
220 LABEL"SFM"
230 'WHILE(INP(&H701) AND &H80):WEND
240 OUT &H700,R
250 OUT &H701,D
260 RETURN
270 '
280 DATA 01,02,01,00     :'TEST
290 DATA 08,00,08,01,08,02,08,03:'KEY OFF (0-3)
300 DATA 08,04,08,05,08,06,08,07:'KEY OFF (4-7)
310 DATA 0F,00000000             :'NE/FREQ
320 DATA 10,00,11,00,12,00,14,00:'CLK A1,A2,B
330 DATA 18,220          :'LFRQ
340 DATA 19,00000100     :'PMD/AMD
350 DATA 1B,00000010     :'CT/W
360 DATA 20,11111010     :'RL/FL/CON
370 DATA 30,00010100     :'KF
380 DATA 38,00010001     :'PMS/AMS
390 DATA 40,01010001     :'DT1/MUL(1)
400 DATA 50,01110001     :'DT1/MUL(2)
410 DATA 48,00100101     :'DT1/MUL(3)
420 DATA 58,00010001     :'DT1/MUL(4)
430 DATA 60,037,70,062,68,077,78,010:'TL(1-4)
440 DATA 80,5F,90,56,88,5D,98,9F:'KS/AR(1-4)
450 DATA A0,05,B0,00,A8,00,B8,87:'AMS-EN/D1R(1-4)
460 DATA C0,07,D0,04,C8,04,D8,06:'DT2/D2R(4)
470 DATA E0,94,F0,45,E8,45,F8,45:'D1L/RR(4)
480 DATA 28,3C,08,78,P9,08,00,P6
490 DATA 28,3E,08,78,P9,08,00,P6
500 DATA 28,41,08,78,P9,08,00,P6
510 DATA 28,42,08,78,P9,08,00,P6
520 DATA 28,45,08,78,P9,08,00,P6
530 DATA 28,48,08,78,P9,08,00,P6
540 DATA 28,4A,08,78,P9,08,00,P6
550 DATA 28,4C,08,78,P9,08,00,P6
560 DATA !
```

リスト 11-2　ドミソ用変更点

```
231 IF (&H28<=R) AND (R<=&H2F) THEN 240
232 IF (R>=&H20) THEN FOR L=0 TO 2:GOSUB240:R=R+1:NEXT:RETURN


480 DATA 28,3C,29,41,2A,45
490 DATA 08,78,P4,08,79,P4,08,7A,P4
500 DATA 08,00,08,01,08,02
510 DATA 08,78,08,79,08,7A,P9
520 DATA 08,00,08,01,08,02,P6
530 DATA !
```

# トーンダイアラである

これから MML の作成に取りかかるわけであるが，その前に FM 音源の，なかなかにその筋な使い方を発見したので紹介しておくのである。それは FM 音源で電話をかけてしまおうというものである。つまりトーンダイアラなわけだ。ただしこれはプッシュホンだけ

に有効なワザである（世の中には一見するとプッシュホンだが，実はダイヤル式の電話を押しボタン式に変えただけのものもある。そのような方式だと使えない)。

解説しよう。プッシュホンは，ボタンを押すとそれに対応した2種類の高さの正弦波が出るような仕組みになっているのである。例の「ピポパピポ」という音である。で，具体的にどのような周波数の音が出ているかというとそれが**図 11-10** である。一目瞭然であろうが，たとえば「5」のボタンを押すと770Hzと1336Hzの二つの正弦波が出るのである。ちなみにみんな知っているように，一般の電話機にはA，B，C，Dのボタンは付いていない。しかしこのように決まっているそうである。

**図 11-10　押しボタンの配列と周波数の関係**

| 低群＼高群 | 1209Hz | 1336Hz | 1477Hz | 1633Hz |
|---|---|---|---|---|
| 697Hz | 1 | 2 | 3 | A |
| 770Hz | 4 | 5 | 6 | B |
| 852Hz | 7 | 8 | 9 | C |
| 941Hz | * | 0 | # | D |

さて，ここからが肝心なのであるが，実はプッシュホンでボタンを押して出る「ピポパピポ」の音も，人間が送話口でしゃべる音も，電話回線にとっては同じなのである。よって，正確な周波数で口笛を吹ける人が2人いて，電話機の送話口に向かって「ピポポパ」と口笛を吹くと，ちゃんと電話がかかってしまうのである。しかし残念なことにそこまで正確に口笛を吹ける人を2人も連れてくることは難しいので，その代わりにFM音源を使ってしまおうというわけである。そしてそれが**リスト 11-3** である。使い方は，

**1)**　正確に打ち込む。

**2)**　110行を取り去り，120行のREM(')を消してRUNする。

**3)**　キーボードの数字キーを押し，プッシュホンのボタンを押したときに聞こえる音と，**できるだけ近くなるように**ボードに付いているボリウムを調節する。

**4)**　電話の受話器を取り，送話口をスピーカに近づけ，ピポパする。ただしこのとき117などのように**緊急電話に近い番号は避ける**。最初は自分の電話番号にかけて，話し中であることを確認するのがよいであろう。もしも間違い電話になってしまったらよく謝ること。

というわけで，この機能を使えばソフトだけで（送話口をスピーカに持ってくる必要はあるが）オートダイヤルできるのである。住所録や電話帳ソフトに期待したい。なお，以上の行動はNTTから怒られることはないはずであるから安心して使っていただきたい

(だってモジュラージャックに直接接続するわけではない)。

ちなみにこのテクニックを応用すれば，ソフトウェアだけで「送信だけの音響カプラ」ができるはずである(きっと信頼性は最低だろう)。ぜひとも自由研究していただきたい。

リスト 11-3　トーンダイアラ

```
100 GOSUB"INIT"
110 D$="604 1111":GOSUB 150
120 'D$=INKEY$(1):GOSUB 150:GOTO120
130 END
140 '
150 FOR I=1 TO LEN(D$)
160   A$=MID$(D$,I,1):PRINT A$;
170   GOSUB200
180 NEXT:RETURN
190 '
200 T=INSTR("123456789*0#",A$):IF T=0 THEN RETURN
210 T=T-1:L=T ¥ 3:H=4+(T MOD 3)
220 R=8:D=&H78+L:GOSUB 380
230 R=8:D=&H78+H:GOSUB 380
240 PAUSE 1
250 R=8:D=L:GOSUB 380
260 R=8:D=H:GOSUB 380
270 RETURN
280 '
290 LABEL"INIT"
300 READ R$:R1$=LEFT$(R$,1):IF R1$="!" THEN RETURN
310 R=VAL("&H"+MID$(R$,1,2))
320 M$=MID$(R$,3,1):IF M$<>"-" THEN 340
330 R0=R:R1=VAL("&H"+MID$(R$,4,2))
340 READ D$:D=VAL("&H"+D$)
350 GOSUB 370:GOTO 300
360 '
370 IF M$="-" THEN FOR R=R0 TO R1:GOSUB380:NEXT:RETURN
380 OUT &H700,R
390 OUT &H701,D
400 RETURN
410 '
420 DATA 01,02,01,00      :'TEST
430 DATA 08,00,08,01,08,02,08,03:'KEY OFF (0-3)
440 DATA 08,04,08,05,08,06,08,07:'KEY OFF (4-7)
450 DATA 0F-1B,00                :'NE/FREQ...
460 DATA 20-27,47         :'RL/FL/CON
470 DATA 28,51,29,54,2A,56,2B,59:'KCODE (0-3)
480 DATA 2C,5E,2D,61,2E,62,2F,00:'KCODE (4-7)
490 DATA 30,EC,31,A8,32,6C,33,24:'KF (0-3)
500 DATA 34,78,35,34,36,EC,37,00:'KF (4-7)
510 DATA 38-3F,00:'PMS/AMS
520 DATA 40-5F,01         :'DT1/MUL
530 DATA 60-7F,20         :'TL
540 DATA 80-9F,1F         :'KS/AR
550 DATA A0-DF,00         :'AMS-EN/1DR,DT2/2DR
560 DATA E0-FF,FF         :'1DL/RR
570 DATA !
```

というところで，FM 音源の具体的な応用として MML（ミュージック・マクロ・ランゲージ：よーするに BASIC の PLAY 文が多少進化したやつ）を作ったりするわけだ。そしてあな恐ろしや，なんと BASIC の書き換えに手を染めてしまったのであった。そう，はっきり言って私は BASIC の書き換えが嫌いなのである。なぜかというと，最大の理由は BASIC というものは結構ちまちまとバグ取りのためにバージョンアップされていたりするからである。大抵の場合はそれでもたいした影響はないのであるが，しかしやはり**そこ**

はかとなく不安が漂ったりするわけである。もしもどれかのバージョンではこのよーな書き換えが許されなかったりしたら困るのである。しかし使い勝手からすると，どうしても BASIC の書き換えが必要なのである。困ったものである。

# MMLである

**リスト 11-4，11-5，11-6** である。説明すると，リスト 11-4 がソースリスト(BASIC を書き換えるルーチンも含んでいる)，リスト 11-5 がそのダンプリスト，リスト 11-6 がサンプル曲である。

さて使い方である。まずは NEW ON &HC000 を実行した後でリスト 11-5 を打ち込み，チェックサムと CRC を確認する（付録 A を使うこと）。X1 ユーザーであればここまでである。もしも turbo ユーザーである場合は，さらに次の 3 か所を変更する。

A965 ：07 07 → A3 1F
A9D9 ：04 07 → A0 1F
A9E8 ：07 07 → A3 1F

SAVEM "MML. OBJ", &HA8B0, &HAFFF でセーブする。

次に使用方法である。CZ-8FB01 V1.0 を起動する。**NEW BASIC ではないし，turbo BASIC でもない。** NEW ON &HB000 を実行した後，LOADM "MML. OBJ" でロードし，CALL &HA8B0 を実行。これで MML が使えるようになる。

リスト 11-4 MML ソースリスト

```
                                .Z80
                                .PHASE  0A8B0H
                        ;
0704                    CTC     EQU     0704H
4000                    WTOP    EQU     4000H
                        ;
A8B0    21 A900                 LD      HL,0A900H
A8B3    22 2DE3                 LD      (2DE3H),HL
A8B6    22 2E11                 LD      (2E11H),HL
A8B9    22 2E13                 LD      (2E13H),HL
                        ;
A8BC    C9                      RET
                        ;
A8BD                    KOKO:   DS      0A900H-KOKO
                        ;
A900    CD 7FD1                 CALL    7FD1H
A903    3A A5DB                 LD      A,(0A5DBH)      ;A=VAR TYPE
A906    E5                      PUSH    HL              ;SAVE TEXT PTR
A907    EB                      EX      DE,HL
A908    FE 03                   CP      03              ;STRING ?
A90A    28 72                   JR      Z,DOSTR         ;GET STRING
                        ;
                        ;ELSE TEMPO
                        ;============== TEMPO =================
A90C    CD 5436         STEMPO: CALL    5436H           ;CSNG
A90F    CD 5BAF                 CALL    5BAFH           ;->INT (->HL)
A912    7C                      LD      A,H
A913    B5                      OR      L
A914    20 4C                   JR      NZ,PRET
                        ;TEMPO 0=INITIALIZE
A916    21 4001         INIT:   LD      HL,WTOP+1
A919    22 A9D4                 LD      (HEAD),HL
A91C    22 A9D6                 LD      (HEAD0),HL
A91F    2B                      DEC     HL
A920    22 A9D2                 LD      (TAIL),HL       ;INIT POINTERS
A923    44                      LD      B,H
A924    4D                      LD      C,L             ;BC=HL
```

```
A925    AF                     XOR   A
A926    ED 79                  OUT   (C),A              ;DUMMY DATA
A928    3E 1E                  LD    A,30               ;TEMPO 120
A92A    32 AE17                LD    (TMPV),A           ;TEMPO AREA
                      ;
A92D    21 AE78                LD    HL,DFLDMY          ;INIT DFLW
A930    11 AE7C                LD    DE,DFLW
A933    01 0024                LD    BC,4*9
A936    ED B0                  LDIR
A938    23                     INC   HL
A939    36 0C                  LD    (HL),12
A93B    2B                     DEC   HL
A93C    01 0008                LD    BC,4*2
A93F    ED B0                  LDIR
                      ;DE=ACTF
A941    EB                     EX    DE,HL    ;HL=ACTF
A942    06 0B                  LD    B,11
A944    36 00         INIT0:   LD    (HL),0   ;KILL
A946    23                     INC   HL
A947    10 FB                  DJNZ  INIT0
                      ;
A949    F3                     DI
A94A    3E C3                  LD    A,0C3H
A94C    32 013C                LD    (13CH),A
A94F    21 A964                LD    HL,QUIET1
A952    22 013D                LD    (13DH),HL
A955    CD 013C                CALL  013CH              ;QUIET
                      ;
A958    CD A9D8                CALL  TOCTC
A95B    21 AA32                LD    HL,IEXEC
A95E    22 005E                LD    (005EH),HL         ;SET VECTOR
A961    FB                     EI
                      ;
A962    E1            PRET:    POP   HL
A963    C9                     RET
                      ;
                      ;STOP SOUND <-PSG MUTE <- "ctrl-D"
A964    01 0707       QUIET1:  LD    BC,CTC+3
A967    3E 03                  LD    A,3
A969    ED 79                  OUT   (C),A    ;RESET CTC
                      ;
A96B    16 07                  LD    D,7
A96D    3E 08                  LD    A,8
A96F    CD ADD5       Q2:      CALL  WOPM
A972    15                     DEC   D
A973    20 FA                  JR    NZ,Q2
A975    CD ADD5                CALL  WOPM     ;D=0
                      ;
A978    01 1C00                LD    BC,1C00H
A97B    C3 013F                JP    013FH    ;PATCH BACK
                      ;
                      ;
                      ;============ DO STRING ==============
A97E    CD 7FC3       DOSTR:   CALL  7FC3H    ;GET REAL ADDR.
A981    B7                     OR    A
A982    28 03                  JR    Z,PRETX  ;LEN=0
A984    CD A99F                CALL  STORE    ;A=LEN
A987    E1            PRETX:   POP   HL
A988    7E                     LD    A,(HL)
A989    D6 3B                  SUB   ';'      ;MUSIC "...";
A98B    23                     INC   HL
A98C    C8                     RET   Z
A98D    2B                     DEC   HL       ;BACK HL
A98E    ED 4B A9D6    PRETXX:  LD    BC,(HEAD0)
A992    AF                     XOR   A
A993    ED 79                  OUT   (C),A    ;SET END MARK
A995    03                     INC   BC
A996    ED 43 A9D6             LD    (HEAD0),BC
A99A    ED 43 A9D4             LD    (HEAD),BC          ;COPY
A99E    C9                     RET
                      ;
A99F    ED 4B A9D6    STORE:   LD    BC,(HEAD0)
A9A3    2A A9D4                LD    HL,(HEAD)
A9A6    B7                     OR    A
A9A7    ED 42                  SBC   HL,BC
A9A9    20 05                  JR    NZ,STORE1
A9AB    2E 3A                  LD    L,':'
A9AD    ED 69                  OUT   (C),L
A9AF    03                     INC   BC
A9B0    6F            STORE1:  LD    L,A                ;L=LEN
A9B1    1A            SLOOP:   LD    A,(DE)
```

```
                                  ;UPPER
A9B2    FE 61                               CP      'a'
A9B4    38 06                               JR      C,S0
A9B6    FE 7B                               CP      'z'+1
A9B8    30 02                               JR      NC,S0
A9BA    C6 E0                               ADD     A,'A'-'a'
                                  ;
A9BC    ED 79                     S0:       OUT     (C),A
A9BE    03                                  INC     BC
A9BF    13                                  INC     DE
                                  ;
A9C0    78                                  LD      A,B
A9C1    B1                                  OR      C
A9C2    28 08                               JR      Z,FULL
A9C4    2D                                  DEC     L
A9C5    20 EA                               JR      NZ,SLOOP
                                  ;
A9C7    ED 43 A9D6                          LD      (HEAD0),BC
A9CB    C9                                  RET
                                  ;
A9CC    CD 013C                   FULL:     CALL    13CH
A9CF    C3 2027                             JP      2027H   ;OUT OF MEM
                                  ;
A9D2    4000                      TAIL:     DEFW    4000H
A9D4    4000                      HEAD:     DEFW    4000H
A9D6    4000                      HEAD0:    DEFW    4000H
                                  ;
A9D8    01 0704                   TOCTC:    LD      BC,CTC
A9DB    3E 27                               LD      A,100111B
A9DD    ED 79                               OUT     (C),A
A9DF    3E 01                               LD      A,1
A9E1    ED 79                               OUT     (C),A
A9E3    3E 58                               LD      A,58H           ;SET VECTOR
A9E5    ED 79                               OUT     (C),A
A9E7    01 0707                   TOCTC1:   LD      BC,CTC+3        ;CH3
A9EA    3E C7                               LD      A,11000111B
A9EC    ED 79                               OUT     (C),A
A9EE    3A AE17                             LD      A,(TMPV)        ;TEMPO
A9F1    ED 79                               OUT     (C),A
A9F3    C9                                  RET
                                  ;
                                  ;============ INT ROUTINE ============
                                  ;
A9F4                                        DS      30*2
AA30                              INTSP:    DS      2
                                  ;
AA32    F3                        IEXEC:    DI
AA33    ED 73 AA30                          LD      (INTSP),SP
AA37    31 AA30                             LD      SP,INTSP
AA3A    F5                                  PUSH    AF
AA3B    C5                                  PUSH    BC
AA3C    D5                                  PUSH    DE
AA3D    E5                                  PUSH    HL
                                  ;
AA3E    21 AEA8                             LD      HL,ACTF
AA41    06 0B                               LD      B,11
AA43    AF                                  XOR     A
AA44    B6                        IE0:      OR      (HL)
AA45    23                                  INC     HL
AA46    20 4A                               JR      NZ,IE1  ;SOMTHING ALIVE
AA48    10 FA                               DJNZ    IE0
                                  ;
                                  ;NEXT BLOCK OR END
AA4A    ED 4B A9D2                          LD      BC,(TAIL)
AA4E    2A A9D4                   IEQ:      LD      HL,(HEAD)
AA51    2B                                  DEC     HL
AA52    B7                                  OR      A
AA53    ED 42                               SBC     HL,BC
AA55    28 51                               JR      Z,IE9   ;EMPTY -> NOTHING
                                  ;
                                  ;NEXT BLOCK AND SET PCS
AA57    ED 78                               IN      A,(C)
AA59    03                                  INC     BC
AA5A    28 0C                               JR      Z,START
                                  ;
AA5C    ED 78                     DROP:     IN      A,(C)
AA5E    03                                  INC     BC
AA5F    20 FB                               JR      NZ,DROP
AA61    0B                                  DEC     BC
AA62    ED 43 A9D2                          LD      (TAIL),BC
AA66    18 E6                               JR      IEQ
```

```
                                    ;
AA68     ED 43 A9D2                 START:  LD      (TAIL),BC
AA6C     21 AEA8                            LD      HL,ACTF
AA6F     E5                                 PUSH    HL
AA70     21 AEB3                            LD      HL,PC
AA73     16 00                              LD      D,0
AA75     ED 78                      SPC0:   IN      A,(C)
AA77     28 05                              JR      Z,SPC1   ;HIT STRING END
AA79     03                                 INC     BC
AA7A     FE 3A                              CP      ':'
AA7C     20 F7                              JR      NZ,SPC0
                                    ;
AA7E     B7                         SPC1:   OR      A
AA7F     28 02                              JR      Z,SPC2
AA81     3E 01                              LD      A,1
AA83     E3                         SPC2:   EX      (SP),HL
AA84     77                                 LD      (HL),A   ;!!
AA85     23                                 INC     HL       ;INC ACTF ADDR.
AA86     E3                                 EX      (SP),HL
AA87     71                                 LD      (HL),C
AA88     23                                 INC     HL
AA89     70                                 LD      (HL),B
AA8A     23                                 INC     HL
AA8B     14                                 INC     D
AA8C     7A                                 LD      A,D
AA8D     FE 0B                              CP      11       ;CHECK COUNTER
AA8F     38 E4                              JR      C,SPC0
AA91     E1                                 POP     HL       ;DROP ACTF
                                    ;
                                    ;PCS ARE READY
                                    ;00H=DEAD,01H=ALIVE,80H=ACTIVE
AA92     11 0000                    IE1:    LD      DE,0     ;CHANNEL
AA95     21 AEA8                            LD      HL,ACTF
AA98     D5                         IEL:    PUSH    DE
AA99     E5                                 PUSH    HL
AA9A     7E                                 LD      A,(HL)
AA9B     B7                                 OR      A
AA9C     C4 AAB3                            CALL    NZ,ACT   ;ALIVE OR ACTIVE
AA9F     E1                                 POP     HL
AAA0     D1                                 POP     DE
AAA1     23                                 INC     HL
AAA2     1C                                 INC     E
AAA3     7B                                 LD      A,E
AAA4     FE 0B                              CP      11
AAA6     20 F0                              JR      NZ,IEL
                                    ;
AAA8     E1                         IE9:    POP     HL
AAA9     D1                                 POP     DE
AAAA     C1                                 POP     BC
AAAB     F1                                 POP     AF
AAAC     ED 7B AA30                         LD      SP,(INTSP)
AAB0     FB                                 EI
AAB1     ED 4D                              RETI             ;RET FROM INT
                                    ;
                                    ;============= SUB ==============
                                    ;
AAB3     E5                         ACT:    PUSH    HL       ;SAVE ACTF
AAB4     D5                                 PUSH    DE       ;SAVE CH NO.
AAB5     F2 AACA                            JP      P,ACT7
                                    ;
AAB8     CD ADE0                            CALL    RCTR     ;READ COUNTER
                                    ;BC=OMOTE,HL=URA
AABB     78                                 LD      A,B
AABC     B1                                 OR      C
AABD     D1                                 POP     DE
AABE     E5                                 PUSH    HL
AABF     CC ACDB                            CALL    Z,KEYOFF
AAC2     E1                                 POP     HL
AAC3     7C                                 LD      A,H
AAC4     B5                                 OR      L
AAC5     E1                                 POP     HL
AAC6     C0                                 RET     NZ
AAC7     36 01                              LD      (HL),1   ;SLEEP
AAC9     C9                                 RET
                                    ;
                                    ;ALIVE AND NOT-ACTIVE
AACA     CB 23                      ACT7:   SLA     E        ;DE=DE*2
AACC     21 AEB3                            LD      HL,PC
AACF     19                                 ADD     HL,DE
AAD0     4E                                 LD      C,(HL)
AAD1     23                                 INC     HL
```

```
AAD2    46                LD      B,(HL)   ;GET PC
                  ;
AAD3    ED 78             IN      A,(C)    ;READ NOTE
AAD5    28 04             JR      Z,DIE
AAD7    FE 3A             CP      ':'
AAD9    20 08             JR      NZ,WAKE  ;HIT NOTE
                  ;
AADB    D1        DIE:    POP     DE
AADC    70        DIE2:   LD      (HL),B
AADD    2B                DEC     HL
AADE    71                LD      (HL),C   ;SAVE PC
                  ;
AADF    E1                POP     HL       ;GET ACTF
AAE0    36 00             LD      (HL),0   ;KILL (ACTF)
AAE2    C9                RET
                  ;
                  ;WAKE UP
AAE3    D1        WAKE:   POP     DE       ;DE=CHANNEL NO.
AAE4    ED 78     WUL:    IN      A,(C)
AAE6    28 F4             JR      Z,DIE2
AAE8    FE 3A             CP      ':'
AAEA    28 F0             JR      Z,DIE2
AAEC    D5                PUSH    DE
AAED    E5                PUSH    HL       ;SAVE PC.ADDR.
AAEE    CD AAFE           CALL    DONOTE   ;DO NOTE
AAF1    E1                POP     HL
AAF2    28 E7             JR      Z,DIE    ;HIT ':' OR 00H
AAF4    D1                POP     DE       ;GET CH NO.
AAF5    38 ED             JR      C,WUL    ;NOT REAL NOTE
                  ;
                  ;HIT REAL NOTE            ;A-G,N,W
AAF7    70                LD      (HL),B
AAF8    2B                DEC     HL
AAF9    71                LD      (HL),C   ;SAVE PC
                  ;
AAFA    E1                POP     HL       ;GET ACTF.ADDR
AAFB    36 80             LD      (HL),80H          ;MAKE ACTIVE
AAFD    C9                RET
                  ;
                  ;============= DO NOTE ==============
                  ;DE=CHANNEL NO. (BREAK OK)
AAFE    ED 78     DONOTE: IN      A,(C)
AB00    03                INC     BC
AB01    FE 41             CP      'A'
AB03    38 2A             JR      C,DN0    ;?<A
AB05    FE 48             CP      'G'+1
AB07    30 34             JR      NC,DN1   ;?>G
                  ;
AB09    CD AB1A           CALL    GETNT    ;GET A-G
                  ;
AB0C    C5        RN0:    PUSH    BC
AB0D    D5                PUSH    DE
AB0E    CD AC67           CALL    KEYON
AB11    D1                POP     DE
AB12    C1                POP     BC
AB13    CD ACED   RN1:    CALL    DONUM    ;NEW BC
AB16    3E 01     RN2:    LD      A,1
AB18    B7                OR      A        ;NZ,NC
AB19    C9                RET

                  ;GET A-G
AB1A    D6 41     GETNT:  SUB     'A'
AB1C    87                ADD     A,A
AB1D    6F                LD      L,A      ;SAVE NOTE
AB1E    ED 78             IN      A,(C)
AB20    03                INC     BC
AB21    FE 2D             CP      '-'
AB23    C8                RET     Z
AB24    2C                INC     L
AB25    2C                INC     L
AB26    FE 2B             CP      '+'
AB28    C8                RET     Z
AB29    FE 23             CP      '#'
AB2B    C8                RET     Z
AB2C    2D                DEC     L        ;NOMAL NOTE
AB2D    0B                DEC     BC
AB2E    C9                RET
                  ;L=CODE,
                  ;
                  ;@@@@@@ &,{}
                  ;@@@@@@
```

```
                                ;&,>,<,
AB2F    FE 3E                   DN0:    CP      '>'     ;OCT UP
AB31    CA AC21                         JP      Z,UPDOWN
AB34    FE 3C                           CP      '<'     ;OCT DOWN
AB36    CA AC21                         JP      Z,UPDOWN
                                ;
AB39    7F                      ACT8:   LD      A,A
AB3A    B7                              OR      A
AB3B    37                              SCF
AB3C    C9                              RET             ;NZ,C

                                ;R,N,Q,L,T,O,V,I,Y,W
AB3D    FE 4E                   DN1:    CP      'N'     ;NOTE BY CODE
AB3F    28 51                           JR      Z,NPLAY
                                ;
AB41    FE 52                           CP      'R'
AB43    28 CE                           JR      Z,RN1   ;Rn
                                ;
AB45    FE 49                           CP      'I'     ;GAKKI BANGOU
AB47    CA AEF5                         JP      Z,INST0
                                ;
AB4A    FE 59                           CP      'Y'     ;0-7:OPM,8-10:PSG
AB4C    CA ABF3                         JP      Z,WREG
                                ;
AB4F    FE 57                           CP      'W'     ;BIG OMOTE,SMALL URA
AB51    CA ABDD                         JP      Z,WAIT
                                ;
AB54    FE 54                           CP      'T'     ;TEMPO
AB56    CA AC33                         JP      Z,STRUN ;RUNTIME SET TEMPO
                                ;
AB59    21 0000                         LD      HL,0
AB5C    FE 4F                           CP      'O'
AB5E    28 0F                           JR      Z,SETGO
AB60    2C                              INC     L
AB61    FE 56                           CP      'V'
AB63    28 0A                           JR      Z,SETGO
AB65    2C                              INC     L
AB66    FE 51                           CP      'Q'     ;RATIO
AB68    28 05                           JR      Z,SETGO
AB6A    2C                              INC     L
AB6B    FE 4C                           CP      'L'     ;DEFAULT LEN
AB6D    20 CA                           JR      NZ,ACT8 ;NEXT
                                ;
AB6F    E5                      SETGO:  PUSH    HL
AB70    CD AE0E                         CALL    DFLTBL
AB73    D1                              POP     DE
AB74    19                              ADD     HL,DE
AB75    E5                              PUSH    HL
AB76    F5                              PUSH    AF
AB77    CD AD90                         CALL    NUMBER
AB7A    30 04                           JR      NC,STG1
AB7C    F1                              POP     AF
AB7D    D1                              POP     DE
AB7E    18 B9                           JR      ACT8
                                ;
AB80    67                      STG1:   LD      H,A     ;SAVE LAST A
AB81    F1                              POP     AF
AB82    D1                              POP     DE
AB83    EB                              EX      DE,HL
AB84    73                              LD      (HL),E
AB85    FE 4C                           CP      'L'
AB87    20 B0                           JR      NZ,ACT8 ;RET WITH C,NZ
AB89    7A                              LD      A,D
AB8A    FE 02                           CP      '0'-'.'
AB8C    20 AB                           JR      NZ,ACT8
AB8E    CB FE                           SET     7,(HL)  ;SET BIT7
AB90    18 A7                           JR      ACT8
                                ;
AB92    D5                      NPLAY:  PUSH    DE
AB93    CD AD90                         CALL    NUMBER
AB96    38 3F                           JR      C,NPLAY9
AB98    7D                              LD      A,L
AB99    FE 60                           CP      96
AB9B    30 3A                           JR      NC,NPLAY9
                                ;A=A MOD 12,E=A/12
AB9D    1E 00                           LD      E,0
AB9F    D6 0C                   NPL:    SUB     12
ABA1    38 03                           JR      C,NPLAY0
ABA3    1C                              INC     E
ABA4    18 F9                           JR      NPL
ABA6    C6 11                   NPLAY0: ADD     A,12+5
```

```
ABA8    FE 0A                   CP      10
ABAA    38 07                   JR      C,NPLAY1
ABAC    3C                      INC     A
ABAD    FE 0F                   CP      15
ABAF    38 02                   JR      C,NPLAY1
ABB1    D6 0E                   SUB     14
ABB3    57              NPLAY1: LD      D,A
ABB4    1C                      INC     E       ;D=CODE,E=OCT
ABB5    EB                      EX      DE,HL
ABB6    D1                      POP     DE
ABB7    D5                      PUSH    DE      ;PUSH CH
ABB8    E5                      PUSH    HL      ;PUSH CODE,OCT
ABB9    CD AE0E                 CALL    DFLTBL
ABBC    22 ABDB                 LD      (NPLAYW),HL
ABBF    D1                      POP     DE
ABC0    7E                      LD      A,(HL)  ;PICK OCT
ABC1    73                      LD      (HL),E  ;SET OCT
                        ;HL=CH NO.
ABC2    6A                      LD      L,D
ABC3    D1                      POP     DE
ABC4    F5                      PUSH    AF
ABC5    C5                      PUSH    BC
ABC6    D5                      PUSH    DE
ABC7    CD AC67                 CALL    KEYON
ABCA    D1                      POP     DE
ABCB    C1                      POP     BC
ABCC    CD ACED                 CALL    DONUM
ABCF    F1                      POP     AF
ABD0    2A ABDB                 LD      HL,(NPLAYW)
ABD3    77                      LD      (HL),A  ;BACK OCT
ABD4    C3 AB16                 JP      RN2
                        ;
ABD7    D1              NPLAY9: POP     DE
ABD8    C3 AB39                 JP      ACT8
                        ;
ABDB                    NPLAYW: DS      2
                        ;
ABDD    D5              WAIT:   PUSH    DE
ABDE    CD ACED                 CALL    DONUM
ABE1    D1                      POP     DE
ABE2    D5                      PUSH    DE
ABE3    C5                      PUSH    BC
ABE4    CD ADE0                 CALL    RCTR
ABE7    23                      INC     HL
ABE8    01 FFFF                 LD      BC,0FFFFH
ABEB    CD ADFC                 CALL    WCTR
ABEE    C1                      POP     BC
ABEF    D1                      POP     DE
ABF0    C3 AB16                 JP      RN2
                        ;
ABF3    D5              WREG:   PUSH    DE
ABF4    CD AD90                 CALL    NUMBER
ABF7    38 24                   JR      C,WREG9
ABF9    ED 78                   IN      A,(C)
ABFB    FE 2C                   CP      ','
ABFD    20 1E                   JR      NZ,WREG9
ABFF    03                      INC     BC
AC00    E5                      PUSH    HL        ;SAVE REG NO.
AC01    CD AD90                 CALL    NUMBER
AC04    7D                      LD      A,L
AC05    E1                      POP     HL
AC06    38 15                   JR      C,WREG9
AC08    D1                      POP     DE
                        ;L=REG NO.,A=DATA,E=CHANNEL
AC09    C5                      PUSH    BC
AC0A    57                      LD      D,A       ;DATA
AC0B    7B                      LD      A,E
AC0C    FE 08                   CP      8
AC0E    7D                      LD      A,L
AC0F    38 07                   JR      C,WROPM
AC11    CD ADC3                 CALL    WPSG
AC14    C1              WREG1:  POP     BC
AC15    C3 AB39                 JP      ACT8
                        ;
                        ;WRITE REGISTER (OPM)
AC18    CD ADD5         WROPM:  CALL    WOPM
AC1B    18 F7                   JR      WREG1
                        ;
AC1D    D1              WREG9:  POP     DE
AC1E    C3 AB39                 JP      ACT8
                        ;
```

```
AC21    D6 3D         UPDOWN: SUB     '<'+1
AC23    CD AE0E               CALL    DFLTBL
AC26    86                    ADD     A,(HL)
AC27    CA AB39               JP      Z,ACT8
AC2A    FE 09                 CP      9
AC2C    D2 AB39               JP      NC,ACT8
AC2F    77                    LD      (HL),A
AC30    C3 AB39               JP      ACT8
                      ;
                      ;TEMPO
AC33    CD AD90       STRUN:  CALL    NUMBER
AC36    DA AB39               JP      C,ACT8
AC39    7D                    LD      A,L
AC3A    FE 1E                 CP      30
AC3C    DA AB39               JP      C,ACT8   ;TOO SMALL
AC3F    C5                    PUSH    BC
                      ;
AC40    4D                    LD      C,L
AC41    06 00                 LD      B,0
AC43    11 0E4E               LD      DE,3662
AC46    21 0000               LD      HL,0
AC49    3E 10                 LD      A,16
AC4B    CB 23         STR1:   SLA     E
AC4D    CB 12                 RL      D
AC4F    ED 6A                 ADC     HL,HL
AC51    B7                    OR      A
AC52    ED 42                 SBC     HL,BC
AC54    30 02                 JR      NC,STR2
AC56    09                    ADD     HL,BC    ;BACK
AC57    1D                    DEC     E
AC58    1C            STR2:   INC     E
AC59    3D                    DEC     A
AC5A    20 EF                 JR      NZ,STR1
                      ;
AC5C    7B                    LD      A,E
AC5D    32 AE17               LD      (TMPV),A
AC60    CD A9E7               CALL    TOCTC1
                      ;
AC63    C1                    POP     BC
AC64    C3 AB39               JP      ACT8
                      ;
                      ;L=CODE (A-,A,A+,.....G,G+):0-14
AC67    7B            KEYON:  LD      A,E      ;CHANNEL NO.
AC68    FE 08                 CP      8
AC6A    38 3A                 JR      C,OPMDO
                      ;
AC6C    7D            PSGDO:  LD      A,L      ;A=CODE
AC6D    D5                    PUSH    DE
AC6E    CD AE0E               CALL    DFLTBL
AC71    46                    LD      B,(HL)   ;GET OCT
AC72    23                    INC     HL
AC73    4E                    LD      C,(HL)   ;VOL
AC74    87                    ADD     A,A
AC75    5F                    LD      E,A      ;D=0
AC76    21 AE18               LD      HL,PSGFRQ
AC79    19                    ADD     HL,DE
AC7A    56                    LD      D,(HL)
AC7B    23                    INC     HL
AC7C    5E                    LD      E,(HL)   ;ED=FREQ
AC7D    CB 3B         PSG1:   SRL     E
AC7F    CB 1A                 RR      D        ;DE=DE/2
AC81    10 FA                 DJNZ    PSG1
                      ;
AC83    E1                    POP     HL       ;HL=CHANNEL NO.
AC84    61                    LD      H,C      ;SAVE VOL
                      ;
                      ;FREQ
AC85    7D                    LD      A,L
AC86    D6 08                 SUB     8        ;A=FREQ LOW
AC88    87                    ADD     A,A
AC89    CD ADC3               CALL    WPSG
AC8C    3C                    INC     A        ;A=FREQ HIGH
AC8D    53                    LD      D,E
AC8E    CD ADC3               CALL    WPSG     ;SET FREQ
                      ;
                      ;VOL
AC91    7D                    LD      A,L      ;CH8 -> R8
AC92    54                    LD      D,H      ;D=VOL
AC93    CD ADC3               CALL    WPSG
                      ;
                      ;TONE?
```

```
AC96    3E 07              LD    A,7
AC98    16 F8              LD    D,0F8H
AC9A    CD ADC3            CALL  WPSG      ;A,B,C=TONE
AC9D    3E 0D              LD    A,13
AC9F    CD ADCC            CALL  RPSG
ACA2    CD ADC3            CALL  WPSG
ACA5    C9                 RET
                   ;
ACA6    7D         OPMDO:  LD    A,L
ACA7    C6 17              ADD   A,17H
ACA9    FE 1B              CP    1BH
ACAB    38 0B              JR    C,OPMDO1
ACAD    D6 10              SUB   10H
ACAF    FE 0F              CP    0FH
ACB1    38 05              JR    C,OPMDO1
ACB3    FE 12              CP    12H
ACB5    28 01              JR    Z,OPMDO1
ACB7    3C                 INC   A
ACB8    D5         OPMDO1: PUSH  DE        ;SAVE CH NO.
ACB9    CD AF52            CALL  OPMV0
ACBC    F5                 PUSH  AF        ;PUSH CODE
ACBD    CD AE0E            CALL  DFLTBL
ACC0    7E                 LD    A,(HL)    ;GET OCT....
ACC1    3D                 DEC   A
ACC2    87                 ADD   A,A
ACC3    87                 ADD   A,A
ACC4    87                 ADD   A,A
ACC5    87                 ADD   A,A
ACC6    57                 LD    D,A
ACC7    F1                 POP   AF
ACC8    82                 ADD   A,D       ;GET CODE
ACC9    57                 LD    D,A
ACCA    E1                 POP   HL
ACCB    7D                 LD    A,L       ;A=CH NO.
ACCC    C6 28              ADD   A,28H
ACCE    CD ADD5            CALL  WOPM
ACD1    7D                 LD    A,L       ;A=CH NO.
ACD2    F6 78              OR    78H
ACD4    57                 LD    D,A
ACD5    3E 08              LD    A,8
ACD7    CD ADD5            CALL  WOPM
ACDA    C9                 RET
                   ;
                   ;E=CHANNEL NO.
ACDB    7B         KEYOFF: LD    A,E
ACDC    FE 08              CP    8
ACDE    38 06              JR    C,OFFOPM         ;OPM
                   ;
ACE0    16 00              LD    D,0
ACE2    CD ADC3            CALL  WPSG      ;VOL=0
ACE5    C9                 RET
                   ;
ACE6    53         OFFOPM: LD    D,E       ;CH NO.
ACE7    3E 08              LD    A,8
ACE9    CD ADD5            CALL  WOPM
ACEC    C9                 RET
                   ;
                   ;E=CHANNEL NO.
ACED    D5         DONUM:  PUSH  DE
ACEE    CD AD04            CALL  DONUMS
                   ;RETURN HL=URA,DE=OMOTE,BC=NEW
ACF1    2B                 DEC   HL
ACF2    1B                 DEC   DE
ACF3    ED 53 AD02         LD    (DONUMW),DE
ACF7    D1                 POP   DE
ACF8    C5                 PUSH  BC
ACF9    ED 4B AD02         LD    BC,(DONUMW)      ;BC=OMOTE
ACFD    CD ADFC            CALL  WCTR
AD00    C1                 POP   BC
AD01    C9                 RET
                   ;
AD02               DONUMW: DS    2
                   ;
AD04    CD AE0E    DONUMS: CALL  DFLTBL
AD07    23                 INC   HL
AD08    23                 INC   HL
AD09    56                 LD    D,(HL)
AD0A    23                 INC   HL
AD0B    5E                 LD    E,(HL)    ;D=Q,E=LEN
AD0C    D5                 PUSH  DE
AD0D    ED 78              IN    A,(C)
```

```
AD0F     FE 40              CP      '@'
AD11     20 21              JR      NZ,DONOML
                   ;DIRECT COUNTER SITEI
AD13     03                 INC     BC
AD14     11 0000            LD      DE,0
                   ;
AD17     CD ADB0   XNUM0:   CALL    DIGIT
AD1A     30 05              JR      NC,XNUMZ
AD1C     11 0001            LD      DE,1
AD1F     18 41              JR      XXX
AD21     5F        XNUMZ:   LD      E,A      ;D=0
AD22     CD ADB0   XNUML:   CALL    DIGIT
AD25     38 3B              JR      C,XXX
AD27     62                 LD      H,D
AD28     6B                 LD      L,E
AD29     29                 ADD     HL,HL
AD2A     29                 ADD     HL,HL
AD2B     19                 ADD     HL,DE
AD2C     29                 ADD     HL,HL
AD2D     5F                 LD      E,A
AD2E     16 00              LD      D,0
AD30     19                 ADD     HL,DE
AD31     EB                 EX      DE,HL
AD32     18 EE              JR      XNUML
                   ;
AD34     CD AD90   DONOML:  CALL    NUMBER   ;
AD37     30 09              JR      NC,DONUM0
                   ;USE DEFAULT VALUE
AD39     6B                 LD      L,E
AD3A     CB 7D              BIT     7,L
AD3C     28 04              JR      Z,DONUM0
                   ;'.'=FUTEN !
AD3E     CB BD              RES     7,L      ;BIT7=0
AD40     3E FE              LD      A,0FEH
AD42     F5        DONUM0:  PUSH    AF
AD43     7D                 LD      A,L
AD44     FE 21              CP      33
AD46     38 02              JR      C,DNOK
AD48     2E 20              LD      L,32     ;MAX 32
AD4A     F1        DNOK:    POP     AF
AD4B     26 00              LD      H,0
AD4D     CB 25              SLA     L        ;HL=HL*2
AD4F     11 AE36            LD      DE,LENTBL
AD52     19                 ADD     HL,DE
AD53     5E                 LD      E,(HL)
AD54     23                 INC     HL
AD55     56                 LD      D,(HL)   ;DE=REAL LEN
                   ;
AD56     FE FE              CP      0FEH
AD58     20 08              JR      NZ,DONUM1
AD5A     62                 LD      H,D
AD5B     6B                 LD      L,E
AD5C     CB 3A              SRL     D
AD5E     CB 1B              RR      E        ;DE=DE/2
AD60     19                 ADD     HL,DE    ;HL=HL*1.5
AD61     EB                 EX      DE,HL
AD62               XXX:
AD62     62        DONUM1:  LD      H,D      ;DE=URA CTR
AD63     6B                 LD      L,E      ;COPY
AD64     F1                 POP     AF       ;A<-D=Q
                   ;KKKK
AD65     E5                 PUSH    HL
AD66     67                 LD      H,A
AD67     ED 78              IN      A,(C)
AD69     FE 26              CP      '&'
AD6B     7C                 LD      A,H
AD6C     E1                 POP     HL
AD6D     20 0D              JR      NZ,DONUM2
                   ;
AD6F     03                 INC     BC
AD70     ED 78              IN      A,(C)
AD72     FE 2B              CP      '+'
AD74     20 18              JR      NZ,DONUM4
AD76     03                 INC     BC
AD77     21 FFFF            LD      HL,0FFFFH
AD7A     18 12              JR      DONUM4
                   ;
AD7C     3D        DONUM2:  DEC     A
AD7D     28 03              JR      Z,DONUM3
AD7F     19                 ADD     HL,DE
AD80     18 FA              JR      DONUM2
```

```
                                        ;
AD82    CB 3C                           DONUM3: SRL     H
AD84    CB 1D                                   RR      L
AD86    CB 3C                                   SRL     H
AD88    CB 1D                                   RR      L
AD8A    CB 3C                                   SRL     H
AD8C    CB 1D                                   RR      L           ;HL=HL/8
                                        ;
                                        ;HL=OMOTE->BC,DE=URA->HL
AD8E    EB                              DONUM4: EX      DE,HL
AD8F    C9                                      RET
                                        ;RETURN HL=URA,DE=OMOTE
                                        ;
                                        ;L=VALUE,Carry=NOTHING,BC=AUTO INC
                                        ;LAST='.' THEN Acc=FEH
AD90    ED 78                           NUMBER: IN      A,(C)
AD92    28 1A                                   JR      Z,NUM9      ;HIT END
AD94    FE 3A                                   CP      ':'
AD96    28 16                                   JR      Z,NUM9      ;HIT END
                                        ;
AD98    CD ADB0                         NUM0:   CALL    DIGIT
AD9B    D8                                      RET     C           ;ERR
                                        ;IKINARI '.' MO ERROR
AD9C    6F                                      LD      L,A
AD9D    CD ADB0                         NUML:   CALL    DIGIT
ADA0    38 0A                                   JR      C,NUM1
ADA2    67                                      LD      H,A         ;SAVE
ADA3    7D                                      LD      A,L
ADA4    87                                      ADD     A,A         ;2
ADA5    87                                      ADD     A,A         ;4
ADA6    85                                      ADD     A,L         ;5
ADA7    87                                      ADD     A,A         ;10
ADA8    84                                      ADD     A,H
ADA9    6F                                      LD      L,A
ADAA    18 F1                                   JR      NUML
                                        ;
ADAC    B7                              NUM1:   OR      A           ;NC
ADAD    C9                                      RET
                                        ;
ADAE    37                              NUM9:   SCF
ADAF    C9                                      RET
                                        ;
ADB0    ED 78                           DIGIT:  IN      A,(C)
ADB2    FE 3A                                   CP      '9'+1
ADB4    38 02                                   JR      C,DIGIT1
ADB6    37                                      SCF
ADB7    C9                                      RET
ADB8    D6 30                           DIGIT1: SUB     '0'
ADBA    03                                      INC     BC
ADBB    D0                                      RET     NC
ADBC    FE FE                                   CP      '.'-'0'
ADBE    28 01                                   JR      Z,DIGIT2
ADC0    0B                                      DEC     BC
ADC1    37                              DIGIT2: SCF
ADC2    C9                                      RET
                                        ;
ADC3    01 1C00                         WPSG:   LD      BC,1C00H
ADC6    ED 79                                   OUT     (C),A
ADC8    05                                      DEC     B
ADC9    ED 51                                   OUT     (C),D
ADCB    C9                                      RET
                                        ;
ADCC    01 1C00                         RPSG:   LD      BC,1C00H
ADCF    ED 79                                   OUT     (C),A
ADD1    05                                      DEC     B
ADD2    ED 50                                   IN      D,(C)
ADD4    C9                                      RET
                                        ;
ADD5    C5                              WOPM:   PUSH    BC
ADD6    01 0700                                 LD      BC,0700H
ADD9    ED 79                                   OUT     (C),A
ADDB    0C                                      INC     C
ADDC    ED 51                                   OUT     (C),D
ADDE    C1                                      POP     BC
ADDF    C9                                      RET
                                        ;
ADE0    D5                              RCTR:   PUSH    DE
ADE1    CB 23                                   SLA     E
ADE3    CB 23                                   SLA     E
ADE5    21 AEC9                                 LD      HL,PACK
ADE8    19                                      ADD     HL,DE
```

```
ADE9     4E                  LD      C,(HL)
ADEA     23                  INC     HL
ADEB     46                  LD      B,(HL)   ;BC=OMOTE
ADEC     23                  INC     HL
ADED     5E                  LD      E,(HL)
ADEE     23                  INC     HL
ADEF     56                  LD      D,(HL)   ;DE=URA
ADF0     EB                  EX      DE,HL    ;HL=URA
ADF1     0B                  DEC     BC
ADF2     2B                  DEC     HL
ADF3     D1                  POP     DE
ADF4     C5                  PUSH    BC
ADF5     E5                  PUSH    HL
ADF6     CD ADFC             CALL    WCTR
ADF9     E1                  POP     HL
ADFA     C1                  POP     BC
ADFB     C9                  RET
                    ;
ADFC     CB 23      WCTR:    SLA     E
ADFE     CB 23               SLA     E
AE00     E5                  PUSH    HL
AE01     21 AEC9             LD      HL,PACK
AE04     19                  ADD     HL,DE
AE05     71                  LD      (HL),C
AE06     23                  INC     HL
AE07     70                  LD      (HL),B   ;BC=OMOTE
AE08     23                  INC     HL
AE09     C1                  POP     BC       ;BC=URA
AE0A     71                  LD      (HL),C
AE0B     23                  INC     HL
AE0C     70                  LD      (HL),B
AE0D     C9                  RET
                    ;
AE0E     21 AE7C    DFLTBL:  LD      HL,DFLW
AE11     CB 23               SLA     E
AE13     CB 23               SLA     E
AE15     19                  ADD     HL,DE
AE16     C9                  RET
                    ;
                    ;============= VAR AREA ===============
                    ;
AE17                TMPV:    DS      1
                    ;
AE18     12D0       PSGFRQ:  DW      12D0H    ;Ab=G#
AE1A     11C2                DW      11C2H    ;A
AE1C     10C4                DW      10C4H    ;A#
AE1E     0FD2                DW      0FD2H    ;B
AE20     0EEE                DW      0EEEH    ;B#=(C)
AE22     1DDE                DW      1DDEH    ;C
AE24     1C2E                DW      1C2EH    ;C#
AE26     1A9A                DW      1A9AH    ;D
AE28     191C                DW      191CH    ;D#
AE2A     17B4                DW      17B4H    ;E
AE2C     165E                DW      165EH    ;E#=F
AE2E     165E                DW      165EH    ;F
AE30     151E                DW      151EH    ;F#
AE32     13EE                DW      13EEH    ;G
AE34     12D0                DW      12D0H    ;G#
                    ;
AE36     0800 0400  LENTBL:  DW      2048,1024,512,341,256
AE3A     0200 0155
AE3E     0100
AE40     00CC 00AA           DW      204,170,146,128,113
AE44     0092 0080
AE48     0071
AE4A     0066 005D           DW      102,93,85,78,73
AE4E     0055 004E
AE52     0049
AE54     0044 0040           DW      68,64,60,56,53
AE58     003C 0038
AE5C     0035
AE5E     0033 0030           DW      51,48,46,44,42
AE62     002E 002C
AE66     002A
AE68     0028 0027           DW      40,39,37,36,35
AE6C     0025 0024
AE70     0023
AE72     0022 0021           DW      34,33,32
AE76     0020
                    ;
                    ;OCT,VOL,Q,LEN::4,100(12),8,4
```

```
AE78    04 64 08 04        DFLDMY: DB      4,100,8,4
AE7C    04 64 08 04        DFLW:   DB      4,100,8,4        ;0
AE80    04 64 08 04                DB      4,100,8,4        ;1
AE84    04 64 08 04                DB      4,100,8,4        ;2
AE88    04 64 08 04                DB      4,100,8,4        ;3
AE8C    04 64 08 04                DB      4,100,8,4        ;4
AE90    04 64 08 04                DB      4,100,8,4        ;5
AE94    04 64 08 04                DB      4,100,8,4        ;6
AE98    04 64 08 04                DB      4,100,8,4        ;7
AE9C    04 0C 08 04                DB      4,12,8,4         ;8
AEA0    04 0C 08 04                DB      4,12,8,4         ;9
AEA4    04 0C 08 04                DB      4,12,8,4         ;10
                           ;
AEA8                       ACTF:   DS      11
AEB3                       PC:     DS      22
AEC9                       PACK:   DS      44
                           ;
                           ;..................
                           ;
                           ;DATA=B000H-B72FH (10*40+36*40)
AEF5    7B                 INST0:  LD      A,E
AEF6    FE 08                      CP      8
AEF8    D2 AB39                    JP      NC,ACT8 ;PSG
AEFB    D5                         PUSH    DE      ;SAVE CH NO.
AEFC    CD AD90                    CALL    NUMBER
AEFF    38 48                      JR      C,INST9
AF01    2D                         DEC     L       ;L=L-1
AF02    7D                         LD      A,L
AF03    FE 28                      CP      40
AF05    30 42                      JR      NC,INST9
                           ;
AF07    D1                         POP     DE
AF08    D5                         PUSH    DE
AF09    21 AFEC                    LD      HL,INSTN
AF0C    19                         ADD     HL,DE
AF0D    77                         LD      (HL),A  ;SAVE INSTN
                           ;
AF0E    6F                         LD      L,A
AF0F    CD AFC8                    CALL    GETVTD
                           ;
                           ;20,38,(40,60,80,A0,C0,E0)*4,(UD+KSL)*4
                           ;KTRNS,18H,PMD,AMD,1BH,DLY+SYNC
AF12    D1                         POP     DE
AF13    C5                         PUSH    BC      ;SAVE PC
AF14    E5                         PUSH    HL      ;SAVE ADDR.
AF15    3E 20                      LD      A,20H
AF17    83                         ADD     A,E
AF18    CD AF4D                    CALL    WOPMX   ;R/L/FB/ALG
AF1B    C6 10                      ADD     A,10H
AF1D    16 14                      LD      D,14H
AF1F    CD ADD5                    CALL    WOPM    ;KF
AF22    C6 08                      ADD     A,08H
AF24    CD AF4D                    CALL    WOPMX   ;PMS/AMS
                           ;
AF27    1E 06                      LD      E,6
AF29    CD AFD7            INST1:  CALL    WOPMXX
AF2C    1D                         DEC     E
AF2D    20 FA                      JR      NZ,INST1
                           ;
AF2F    11 0005                    LD      DE,5
AF32    19                         ADD     HL,DE   ;HL=HL+5
AF33    3E 18                      LD      A,18H
AF35    CD AF4D                    CALL    WOPMX   ;LFRQ
AF38    3C                         INC     A       ;A=19H
AF39    CD AF4D                    CALL    WOPMX   ;PMD
AF3C    CD AF4D                    CALL    WOPMX   ;AMD
AF3F    3E 1B                      LD      A,1BH
AF41    CD AF4D                    CALL    WOPMX   ;FRM
AF44    E1                         POP     HL
AF45    C1                         POP     BC
AF46    C3 AB39                    JP      ACT8
                           ;
AF49    D1                 INST9:  POP     DE      ;DROP
AF4A    C3 AB39                    JP      ACT8
                           ;
AF4D    56                 WOPMX:  LD      D,(HL)
AF4E    23                         INC     HL
AF4F    C3 ADD5                    JP      WOPM
                           ;
AF52    F5                 OPMV0:  PUSH    AF      ;A=CODE
AF53    D5                         PUSH    DE      ;DE=CH NO.
```

```
AF54    D5                      PUSH    DE
AF55    CD AE0E                 CALL    DFLTBL
AF58    23                      INC     HL
AF59    7E                      LD      A,(HL)   ;OVQL
AF5A    D1                      POP     DE
AF5B    F5                      PUSH    AF       ;SAVE V
AF5C    21 AFEC                 LD      HL,INSTN
AF5F    19                      ADD     HL,DE
AF60    6E                      LD      L,(HL)   ;L=In
                        ;
AF61    CD AFC8                 CALL    GETVTD   ;GET ADDR.
AF64    7E                      LD      A,(HL)
AF65    E6 07                   AND     7        ;GET ALG
AF67    0E 07                   LD      C,0111B  ;BIT3=0
AF69    FE 04                   CP      4
AF6B    38 0B                   JR      C,OPMV1  ;ALG0-3
AF6D    CB 89                   RES     1,C      ;OP2=CRY
AF6F    28 07                   JR      Z,OPMV1
AF71    CB 91                   RES     2,C      ;OP3=CRY
AF73    FE 07                   CP      7
AF75    20 01                   JR      NZ,OPMV1
AF77    0D                      DEC     C        ;BIT0=0
                        ;
AF78    11 0006         OPMV1:  LD      DE,6     ;C=MASK
AF7B    19                      ADD     HL,DE    ;POINT V
AF7C    11 AFC4                 LD      DE,V4
AF7F    06 04                   LD      B,4
AF81    7E              OPMV2:  LD      A,(HL)
AF82    CB 09                   RRC     C
AF84    30 02                   JR      NC,OPMV3
AF86    F6 80                   OR      80H      ;MODULE MARK
AF88    12              OPMV3:  LD      (DE),A
AF89    23                      INC     HL
AF8A    13                      INC     DE
AF8B    10 F4                   DJNZ    OPMV2
                        ;
AF8D    06 04                   LD      B,4
AF8F    EB                      EX      DE,HL
AF90    2B                      DEC     HL       ;HL=V4+3
AF91    3E 7F                   LD      A,07FH
AF93    BE              OPMV4:  CP      (HL)
AF94    38 01                   JR      C,OPMV5
                        ;A >= (HL)
AF96    7E                      LD      A,(HL)
AF97    2B              OPMV5:  DEC     HL
AF98    10 F9                   DJNZ    OPMV4
                        ;A=MIN TL
                        ;ADD TO ALL CRY '127-V-A',MIN=0,MAX=7F
AF9A    C1                      POP     BC       ;B=V
AF9B    ED 44                   NEG
AF9D    90                      SUB     B
AF9E    C6 7F                   ADD     A,127
AFA0    4F                      LD      C,A
                        ;
AFA1    D1                      POP     DE
AFA2    D5                      PUSH    DE       ;E=CH NO.
AFA3    3E 58                   LD      A,60H-8
AFA5    83                      ADD     A,E
AFA6    F5                      PUSH    AF       ;SAVE REG#-8
                        ;
AFA7    23                      INC     HL       ;HL=V4
AFA8    E5                      PUSH    HL       ;SAVE V4
AFA9    06 04                   LD      B,4
AFAB    7E              OPMV6:  LD      A,(HL)
AFAC    B7                      OR      A
AFAD    FA AFB6                 JP      M,OPMV7  ;MUDULE
AFB0    81                      ADD     A,C
AFB1    F2 AFB6                 JP      P,OPMV7  ;0-7FH
AFB4    3E 7F                   LD      A,7FH    ;MAX=7FH
AFB6    E6 7F           OPMV7:  AND     7FH
AFB8    77                      LD      (HL),A
AFB9    23                      INC     HL
AFBA    10 EF                   DJNZ    OPMV6
                        ;
AFBC    E1                      POP     HL       ;HL=V4
AFBD    F1                      POP     AF       ;A=REG#-8
AFBE    CD AFD7                 CALL    WOPMXX
                        ;
AFC1    D1                      POP     DE
AFC2    F1                      POP     AF
AFC3    C9                      RET
```

```
                                        ;
AFC4                                    V4:     DS      4
                                        ;
AFC8    26 00                           GETVTD: LD      H,0
AFCA    29                                      ADD     HL,HL   ;2
AFCB    29                                      ADD     HL,HL   ;4
AFCC    54                                      LD      D,H
AFCD    5D                                      LD      E,L     ;COPY
AFCE    29                                      ADD     HL,HL   ;8
AFCF    29                                      ADD     HL,HL   ;16
AFD0    29                                      ADD     HL,HL   ;32
AFD1    19                                      ADD     HL,DE   ;36
AFD2    11 B190                                 LD      DE,0B190H
AFD5    19                                      ADD     HL,DE   ;GET VTD!
AFD6    C9                                      RET
                                        ;
AFD7    C6 08                           WOPMXX: ADD     A,08H
AFD9    CD AF4D                                 CALL    WOPMX
AFDC    C6 10                                   ADD     A,10H
AFDE    CD AF4D                                 CALL    WOPMX
AFE1    D6 08                                   SUB     08H
AFE3    CD AF4D                                 CALL    WOPMX
AFE6    C6 10                                   ADD     A,10H
AFE8    CD AF4D                                 CALL    WOPMX
AFEB    C9                                      RET
                                        ;
AFEC    00 00 00 00                     INSTN:  DB      0,0,0,0
AFF0    01 01 01 01                             DB      1,1,1,1
                                        ;
                                                END
```

リスト 11-5　MML ダンプリスト

```
A8B0 21 00 A9 22 E3 2D 22 11 : 2F
A8B8 2E 22 13 2E C9 4E 41 54 : 3D
A8C0 48 41 4E 49 45 4C 20 53 : 24
A8C8 54 49 54 54 1A 1A 1A 1A : AD
A8D0 1A 1A 1A 1A 1A 1A 1A 1A : D0
A8D8 1A 1A 1A 1A 1A 1A 1A 1A : D0
A8E0 1A 1A 1A 1A 1A 1A 1A 1A : D0
A8E8 1A 1A 1A 1A 1A 1A 1A 1A : D0
A8F0 1A 1A 1A 1A 1A 1A 1A 1A : D0
A8F8 1A 1A 1A 1A 1A 1A 1A 1A : D0
A900 CD D1 7F 3A DB A5 E5 EB : A7
A908 FE 03 28 72 CD 36 54 CD : BF
A910 AF 5B 7C B5 20 4C 21 01 : C9
A918 40 22 D4 A9 22 D6 A9 2B : AB
A920 22 D2 A9 44 4D AF ED 79 : 43
A928 3E 1E 32 17 AE 21 78 AE : 9A
---------------------------------
SUM: A1 89 CC EE 8C 4A A1 79 59F4

A930 11 7C AE 01 24 00 ED B0 : FD
A938 23 36 0C 2B 01 08 00 ED : 86
A940 B0 EB 06 0B 36 00 23 10 : 15
A948 FB F3 3E C3 32 3C 01 21 : 7F
A950 64 A9 22 3D 01 CD 3C 01 : 77
A958 CD D8 A9 21 32 AA 22 5E : CB
A960 00 FB E1 C9 01 07 07 3E : F2
A968 03 ED 79 16 07 3E 08 CD : 99
A970 D5 AD 15 20 FA CD D5 AD : 00
A978 01 00 1C C3 3F 01 CD C3 : B0
A980 7F B7 28 03 CD 9F A9 E1 : 57
A988 7E D6 3B 23 C8 2B ED 4B : DD
A990 D6 A9 AF ED 79 03 ED 43 : C7
A998 D6 A9 ED 43 D4 A9 C9 ED : E2
A9A0 4B D6 A9 2A D4 A9 B7 ED : 15
A9A8 42 20 05 2E 3A ED 69 03 : 28
---------------------------------
SUM: 1F 7B 01 C8 F1 DA 8C F4 7EDA
```

```
A9B0 6F 1A FE 61 38 06 FE 7B : 9F
A9B8 30 02 C6 E0 ED 79 03 13 : 54
A9C0 78 B1 28 08 2D 20 EA ED : 7D
A9C8 43 D6 A9 C9 CD 3C 01 C3 : 58
A9D0 27 20 00 40 00 40 00 40 : 07
A9D8 01 04 07 3E 27 ED 79 3E : 15
A9E0 01 ED 79 3E 58 ED 79 01 : 64
A9E8 07 07 3E C7 ED 79 3A 17 : CA
A9F0 AE ED 79 C9 0A 09 49 4E : 87
A9F8 43 09 44 45 0D 0A 09 44 : 39
AA00 4A 4E 5A 09 4F 50 4D 56 : 3D
AA08 32 CA 03 FF 03 00 00 00 : 01
AA10 00 00 00 00 00 00 00 00 : 00
AA18 00 00 00 00 00 00 00 00 : 00
AA20 00 00 00 00 00 00 00 CA : CA
AA28 56 D9 00 00 00 00 C3 BE : B0
---------------------------------
SUM: 4D A2 6D AB F4 D1 7A 44 C55D

AA30 39 04 F3 ED 73 30 AA 31 : 9B
AA38 30 AA F5 C5 D5 E5 21 A8 : 17
AA40 AE 06 0B AF B6 23 20 4A : B1
AA48 10 FA ED 4B D2 A9 2A D4 : BB
AA50 A9 2B B7 ED 42 28 51 ED : 20
AA58 78 03 28 0C ED 78 03 20 : 37
AA60 FB 0B ED 43 D2 A9 18 E6 : AF
AA68 ED 43 D2 A9 21 A8 AE E5 : 07
AA70 21 B3 AE 16 00 ED 78 28 : 25
AA78 05 03 FE 3A 20 F7 B7 28 : 36
AA80 02 3E 01 E3 77 23 E3 71 : 12
AA88 23 70 23 14 7A FE 0B 38 : 85
AA90 E4 E1 11 00 00 21 A8 AE : 4D
AA98 D5 E5 7E B7 C4 B3 AA E1 : F1
AAA0 D1 23 1C 7B FE 0B 20 F0 : A4
AAA8 E1 D1 C1 F1 ED 7B 30 AA : A6
---------------------------------
SUM: E6 48 BA FB B2 31 EE F1 45E6
```

```
AAB0  FB  ED  4D  E5  D5  F2  CA  AA  :  55
AAB8  CD  E0  AD  78  B1  D1  E5  CC  :  05
AAC0  DB  AC  E1  7C  B5  E1  C0  36  :  70
AAC8  01  C9  CB  23  21  B3  AE  19  :  53
AAD0  4E  23  46  ED  78  28  04  FE  :  46
AAD8  3A  20  08  D1  70  2B  71  E1  :  20
AAE0  36  00  C9  D1  ED  78  28  F4  :  51
AAE8  FE  3A  28  F0  D5  E5  CD  FE  :  D5
AAF0  AA  E1  28  E7  D1  38  ED  70  :  00
AAF8  2B  71  E1  36  80  C9  ED  78  :  61
AB00  03  FE  41  38  2A  FE  48  30  :  1A
AB08  34  CD  1A  AB  C5  D5  CD  67  :  94
AB10  AC  D1  C1  CD  ED  AC  3E  01  :  E3
AB18  B7  C9  D6  41  87  6F  ED  78  :  F2
AB20  03  FE  2D  C8  2C  2C  FE  2B  :  77
AB28  C8  FE  23  C8  2D  0B  C9  FE  :  B0
-------------------------------------------
SUM:  9A  72  30  19  13  2D  68  B7  AC0B

AB30  3E  CA  21  AC  FE  3C  CA  21  :  FA
AB38  AC  7F  B7  37  C9  FE  4E  28  :  56
AB40  51  FE  52  28  CE  FE  49  CA  :  A8
AB48  F5  AE  FE  59  CA  F3  AB  FE  :  60
AB50  57  CA  DD  AB  FE  54  CA  33  :  F8
AB58  AC  21  00  00  FE  4F  28  0F  :  51
AB60  2C  FE  56  28  0A  2C  FE  51  :  2D
AB68  28  05  2C  FE  4C  20  CA  E5  :  72
AB70  CD  0E  AE  D1  19  E5  F5  CD  :  1A
AB78  90  AD  30  04  F1  D1  18  B9  :  04
AB80  67  F1  D1  EB  73  FE  4C  20  :  F1
AB88  B0  7A  FE  02  20  AB  CB  FE  :  BE
AB90  18  A7  D5  CD  90  AD  38  3F  :  15
AB98  7D  FE  60  30  3A  1E  00  D6  :  39
ABA0  0C  38  03  1C  18  F9  C6  11  :  4B
ABA8  FE  0A  38  07  3C  FE  0F  38  :  C8
-------------------------------------------
SUM:  9A  F0  A4  17  6C  3B  F7  8B  113D

ABB0  02  D6  0E  57  1C  EB  D1  D5  :  EA
ABB8  E5  CD  0E  AE  22  DB  AB  D1  :  E7
ABC0  7E  73  6A  D1  F5  C5  D5  CD  :  88
ABC8  67  AC  D1  C1  CD  ED  AC  F1  :  FC
ABD0  2A  DB  AB  77  C3  16  AB  D1  :  7C
ABD8  C3  39  AB  0A  53  D5  CD  ED  :  93
ABE0  AC  D1  D5  C5  CD  E0  AD  23  :  94
ABE8  01  FF  FF  CD  FC  AD  C1  D1  :  07
ABF0  C3  16  AB  D5  CD  90  AD  38  :  9B
ABF8  24  ED  78  FE  2C  20  1E  03  :  F4
AC00  E5  CD  90  AD  7D  E1  38  15  :  9A
AC08  D1  C5  57  7B  FE  08  7D  38  :  23
AC10  07  CD  C3  AD  C1  C3  39  AB  :  AC
AC18  CD  D5  AD  18  F7  D1  C3  39  :  2B
AC20  AB  D6  3D  CD  0E  AE  86  CA  :  97
AC28  39  AB  FE  09  D2  39  AB  77  :  18
-------------------------------------------
SUM:  BB  5E  36  40  EB  04  90  C3  540F

AC30  C3  39  AB  CD  90  AD  DA  39  :  C4
AC38  AB  7D  FE  1E  DA  39  AB  C5  :  C7
AC40  4D  06  00  11  4E  0E  21  00  :  E1
AC48  00  3E  10  CB  23  CB  12  ED  :  06
AC50  6A  B7  ED  42  30  02  09  1D  :  A8
AC58  1C  3D  20  EF  7B  32  17  AE  :  DA
AC60  CD  E7  A9  C1  C3  39  AB  7B  :  40
AC68  FE  08  38  3A  7D  D5  CD  0E  :  A5
AC70  AE  46  23  4E  87  5F  21  18  :  84
AC78  AE  19  56  23  5E  CB  3B  CB  :  6F
AC80  1A  10  FA  E1  61  7D  D6  08  :  C1
AC88  87  CD  C3  AD  3C  53  CD  C3  :  E3
```

```
AC90  AD  7D  54  CD  C3  AD  3E  07  :  00
AC98  16  F8  CD  C3  AD  3E  0D  CD  :  63
ACA0  CC  AD  CD  C3  AD  C9  7D  C6  :  C2
ACA8  17  FE  1B  38  0B  D6  10  FE  :  57
-------------------------------------------
SUM:  AF  39  E6  7D  70  85  27  85  4F0B

ACB0  0F  38  05  FE  12  28  01  3C  :  C1
ACB8  D5  CD  52  AF  F5  CD  0E  AE  :  21
ACC0  7E  3D  87  87  87  87  57  F1  :  1F
ACC8  82  57  E1  7D  C6  28  CD  D5  :  C7
ACD0  AD  7D  F6  78  57  3E  08  CD  :  02
ACD8  D5  AD  C9  7B  FE  08  38  06  :  0A
ACE0  16  00  CD  C3  AD  C9  53  3E  :  AD
ACE8  08  CD  D5  AD  C9  D5  CD  04  :  C6
ACF0  AD  2B  1B  ED  53  02  AD  D1  :  B3
ACF8  C5  ED  4B  02  AD  CD  FC  AD  :  22
AD00  C1  C9  4F  20  CD  0E  AE  23  :  A5
AD08  23  56  23  5E  D5  ED  78  FE  :  32
AD10  40  20  21  03  11  00  00  CD  :  62
AD18  B0  AD  30  05  11  01  00  18  :  BC
AD20  41  5F  CD  B0  AD  38  3B  62  :  9F
AD28  6B  29  29  19  29  5F  16  00  :  74
-------------------------------------------
SUM:  76  1C  3F  52  B9  EA  B3  AB  3973

AD30  19  EB  18  EE  CD  90  AD  30  :  44
AD38  09  6B  CB  7D  28  04  CB  BD  :  70
AD40  3E  FE  F5  7D  FE  21  38  02  :  07
AD48  2E  20  F1  26  00  CB  25  11  :  66
AD50  36  AE  19  5E  23  56  FE  FE  :  D0
AD58  20  08  62  6B  CB  3A  CB  1B  :  E0
AD60  19  EB  62  6B  F1  E5  67  ED  :  FB
AD68  78  FE  26  7C  E1  20  0D  03  :  29
AD70  ED  78  FE  2B  20  18  03  21  :  EA
AD78  FF  FF  18  12  3D  28  03  19  :  A9
AD80  18  FA  CB  3C  CB  1D  CB  3C  :  08
AD88  CB  1D  CB  3C  CB  1D  EB  C9  :  8B
AD90  ED  78  28  1A  FE  3A  28  16  :  1D
AD98  CD  B0  AD  D8  6F  CD  B0  AD  :  9B
ADA0  38  0A  67  7D  87  87  85  87  :  40
ADA8  84  6F  18  F1  B7  C9  37  C9  :  7C
-------------------------------------------
SUM:  BA  42  CC  D3  51  E6  62  5B  A267

ADB0  ED  78  FE  3A  38  02  37  C9  :  D7
ADB8  D6  30  03  D0  FE  FE  28  01  :  FE
ADC0  0B  37  C9  01  00  1C  ED  79  :  8E
ADC8  05  ED  51  C9  01  00  1C  ED  :  16
ADD0  79  05  ED  50  C9  C5  01  00  :  4A
ADD8  07  ED  79  0C  ED  51  C1  C9  :  41
ADE0  D5  CB  23  CB  23  21  C9  AE  :  49
ADE8  19  4E  23  46  23  5E  23  56  :  CA
ADF0  EB  0B  2B  D1  C5  E5  CD  FC  :  65
ADF8  AD  E1  C1  C9  CB  23  CB  23  :  F4
AE00  E5  21  C9  AE  19  71  23  70  :  9A
AE08  23  C1  71  23  70  C9  21  7C  :  4E
AE10  AE  CB  23  CB  23  19  C9  33  :  9F
AE18  D0  12  C2  11  C4  10  D2  0F  :  6A
AE20  EE  0E  DE  1D  2E  1C  9A  1A  :  F5
AE28  1C  19  B4  17  5E  16  5E  16  :  E8
-------------------------------------------
SUM:  69  A9  64  BC  BF  4E  85  7A  0F37

AE30  1E  15  EE  13  D0  12  00  08  :  1E
AE38  00  04  00  02  55  01  00  01  :  5D
AE40  CC  00  AA  00  92  00  80  00  :  88
AE48  71  00  66  00  5D  00  55  00  :  89
AE50  4E  00  49  00  44  00  40  00  :  1B
```

```
AE58 3C 00 38 00 35 00 33 00 : DC
AE60 30 00 2E 00 2C 00 2A 00 : B4
AE68 28 00 27 00 25 00 24 00 : 98
AE70 23 00 22 00 21 00 20 00 : 86
AE78 04 64 08 04 04 64 08 04 : E8
AE80 04 64 08 04 04 64 08 04 : E8
AE88 04 64 08 04 04 64 08 04 : E8
AE90 04 64 08 04 04 64 08 04 : E8
AE98 04 64 08 04 04 0C 08 04 : 90
AEA0 04 0C 08 04 04 0C 08 04 : 38
AEA8 09 4C 44 09 42 43 2C 43 : 96
--------------------------------
SUM: 81 65 6A 36 59 FE 12 64 A752

AEB0 54 43 2B 33 0D 0A 09 4C : 61
AEB8 44 09 41 2C 33 0D 0A 09 : 0D
AEC0 4F 55 54 09 28 43 29 2C : C1
AEC8 41 09 3B 52 45 53 45 54 : 08
AED0 20 43 54 43 0D 0A 3B 0D : 59
AED8 0A 09 4C 44 09 44 2C 37 : 53
AEE0 0D 0A 09 4C 44 09 41 2C : 26
AEE8 38 0D 0A 51 32 3A 09 43 : 58
AEF0 41 4C 4C 09 57 7B FE 08 : BA
AEF8 D2 39 AB D5 CD 90 AD 38 : CD
AF00 48 2D 7D FE 28 30 42 D1 : 5B
AF08 D5 21 EC AF 19 77 6F CD : 5D
AF10 C8 AF D1 C5 E5 3E 20 83 : D3
AF18 CD 4D AF C6 10 16 14 CD : 96
AF20 D5 AD C6 08 CD 4D AF 1E : 37
AF28 06 CD D7 AF 1D 20 FA 11 : A1
--------------------------------
SUM: 37 56 2B AB 7D B1 6B E5 2AF1
```

```
AF30 05 00 19 3E 18 CD 4D AF : 3D
AF38 3C CD 4D AF CD 4D AF 3E : 0C
AF40 1B CD 4D AF E1 C1 C3 39 : 82
AF48 AB D1 C3 39 AB 56 23 C3 : 5F
AF50 D5 AD F5 D5 D5 CD 0E AE : AA
AF58 23 7E D1 F5 21 EC AF 19 : 3C
AF60 6E CD C8 AF 7E E6 07 0E : 2B
AF68 07 FE 04 38 0B CB 89 28 : C8
AF70 07 CB 91 FE 07 20 01 0D : 96
AF78 11 06 00 19 11 C4 AF 06 : BA
AF80 04 7E CB 09 30 02 F6 80 : FE
AF88 12 23 13 10 F4 06 04 EB : 41
AF90 2B 3E 7F BE 38 01 7E 2B : 88
AF98 10 F9 C1 ED 44 90 C6 7F : D0
AFA0 4F D1 D5 3E 58 83 F5 23 : 26
AFA8 E5 06 04 7E B7 FA B6 AF : 83
--------------------------------
SUM: 11 E1 90 1D B7 95 C8 E0 74FD

AFB0 81 F2 B6 AF 3E 7F E6 7F : FA
AFB8 77 23 10 EF E1 F1 CD D7 : 0F
AFC0 AF D1 F1 C9 0D 0A 50 52 : F3
AFC8 26 00 29 29 54 5D 29 29 : 7B
AFD0 29 19 11 90 B1 19 C9 C6 : 3C
AFD8 08 CD 4D AF C6 10 CD 4D : C1
AFE0 AF D6 08 CD 4D AF C6 10 : 2C
AFE8 CD 4D AF C9 00 00 00 00 : 92
AFF0 01 01 01 01             : 04
--------------------------------
SUM: 7B F0 F6 66 44 AF 88 F4 6E00
```

リスト 11-6 サンプル曲(美しく青きドナウ)

```
100 TEMPO 0
110 MUSIC "I1 V127 Q4:I1 V127 Q4:I1 V127 Q4"
120 MUSIC "T150 O4 L4 C&E&G G2>G GRE E<RC C&E&G";
130 MUSIC ":O3 L4      RRR    CGG   CGG CGG   CGG   ";
140 MUSIC ":O3 L4      RRR    REE   REE REE   REE   "
150 MUSIC " G2>G     GRF    FRO3B B&O4D&A A2>A";
160 MUSIC ":O2B>GG< B>GG< B>GG  DGG<     B>GG";
170 MUSIC ":O3RFF    RFF    RFF    RFF       RFF"
180 MUSIC " ARF      FRO3B B&O4D&A A2>A ARE";
190 MUSIC ":O2B>GG< B>GG  DGG        CGG  CGG";
200 MUSIC ":O3RFF    RFF    RFF       REE  REE"
210 MUSIC " ERO4C C&E&G >C2>C CR<G GRO4C C&E&G";
220 MUSIC ":O3CGG CGG    CGG    CGG   CGG    CGG";
230 MUSIC ":O3REE REE    REE    REE   REE    REE"
240 MUSIC " O5C2>C CR<A ARD D&F&A A1       F#G";
250 MUSIC ":O3DAA  DAA  ARR D&F&A <B>GG< B>GG";
260 MUSIC ":O3RFF  RFF  FRR RRR    RFF     RFF"
270 MUSIC " O6E1   C<E E2&D A2&G C1  RR";
280 MUSIC ":O3CGG CGG A2.  G2.  CGG CRR";
290 MUSIC ":O3REE REE F2.  F2.  REE"
```

# MMLの機能である

では次にMMLの機能を説明する。

主な機能はMZ-2500のBASICなどとほぼ同じであるが，若干機能を削ってある。しかし，実害はないはずである（と突っ張る）。削られた機能のうち一番ナニなのが「連符」

の処理である。MZ-2500 では，

PLAY "{CDE}4"

などとすることによって3連符(n連符も可能)が使えたのであるが，こっちではこの機能は使えない。ただしちゃんと逃げ道は用意しておいたので，連符を使いたい人は後で紹介する別の方法を使っていただきたい。

さて機能をまとめたのが**表 11-9** である。一つ大事な点は，音長の指定方法である。HuBASIC では 0 が 32 分音符，1 が 16 分音符，……，9 が全音符となっていたが，ここでは MZ-2500 と同じく，32 が 32 分音符，16 が 16 分音符，……，1 が全音符としてある。なお MZ-2500 にはないが 0 は全音符の 2 倍の長さである。それから，「.」を付けることによって音長は 1.5 倍になる。すなわち「C4.」は符点 4 分音符なわけだ。

**表 11-9　MML の機能表**

| 文　字 | 機　能 |
|---|---|
| C～G，A，B | 音階 |
| ＋，♯ | 音階を半音上げる |
| － | 音階を半音下げる |
| R ℓ | 休符（ℓ：0～32） |
| . | 音長を 1.5 倍にする |
| Nx | 音階コード x による音の発生（x：0～95） |
| Qn | 音の出る時間比率の設定（n：1～8） |
| &(&＋) | 前後の音をつなぐ（タイ）注) |
| L ℓ | 省略時の音長指定（ℓ：0～32） |
| Ts | テンポ指定（s：30～255） |
| On | オクターブ指定（n：1～8） |
| > | オクターブを一つ上げる |
| < | オクターブを一つ下げる |
| Vn | 音量設定（n：0～127 もしくは 0～15） |
| In | FM 音源の音色設定（n：1～40） |
| Yr，d | レジスタ r にデータ d をセット |
| W ℓ | ℓ の長さだけ状態を維持する（ℓ：1～32） |

数値の指定方法
1）"C16" など
2）音長を直接カウンタ値で指定する場合に限り
"C@256" など
注）C2&＋C4＝C2. などとなる

次に指摘しておくのが音階の指定方法である。HuBASIC では「♯C」などで半音上の音を指定していたが，ここでは「C♯」もしくは「C＋」である。半音下の音は「C－」などである。

あと言っておくべきことは「Y」によるレジスタへの直接書き込みである。チャンネル 0～7 であれば OPM，8～10 であれば PSG のレジスタに書き込まれる。

また，1 オクターブ上下の指定であるが，「>」が上がる，「<」が下がるとなっている。

それから，音長（および一般的な数値の指定）には 2 とおりある。

**1）**「C16」，「T120」，「V15」のように，素直に書く方法。このとき数値は 0～255 の範囲である

**2）**「C@512」のように「@」の後に続けて数値を書く。この場合の範囲は 0～65535 で

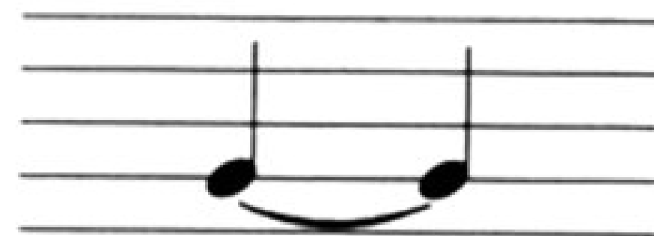
ある。

**3)**の方法は連符の処理のために付けたものである。**表 11-10** を見ていただきたい。これは各音長の長さの内訳である。たとえば「C4」は「C@256」としてもまったく同じ長さの音なのである。よって 4 分音符を 3 個に分けた 3 連符を鳴らしたい場合は 256＝85＋85＋86 より，「C@85 C@85 C@86」とすればよいのである。最後だけちょいと長めだが，まあ人間の耳には分かるまい。

**表 11-10　音調の内訳**

| 音符長 | カウンタ | 音符長 | カウンタ |
|---|---|---|---|
| 0 | 2048 | 17 | 60 |
| 1 | 1024 | 18 | 56 |
| 2 | 512 | 19 | 53 |
| 3 | 341 | 20 | 51 |
| 4 | 256 | 21 | 48 |
| 5 | 204 | 22 | 46 |
| 6 | 170 | 23 | 44 |
| 7 | 146 | 24 | 42 |
| 8 | 128 | 25 | 40 |
| 9 | 113 | 26 | 39 |
| 10 | 102 | 27 | 37 |
| 11 | 93 | 28 | 36 |
| 12 | 85 | 29 | 35 |
| 13 | 78 | 30 | 34 |
| 14 | 73 | 31 | 33 |
| 15 | 68 | 32 | 32 |
| 16 | 64 | | |

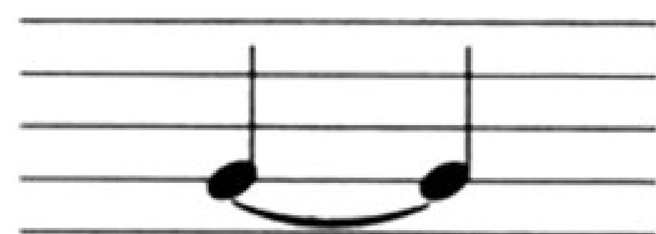
そして私がタコなためにしてしまったことであるが，「タイ」の指定に「&」と「&＋」の 2 とおりがある。



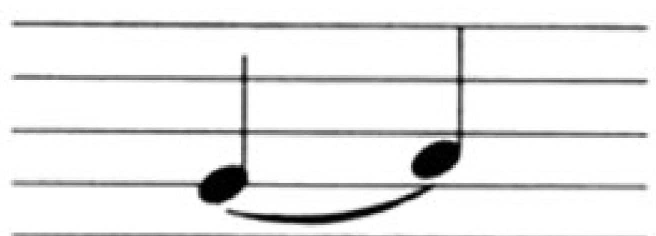
などという符は，「G4&G4」となり，これは「G2」と同じであるべきだったのに，私はこらへんの人情の機微を把握していなかったので，

&＝直前の音符を Q8 で鳴らすだけ

だと思ってしまったのである。これは非常にまずかった。で，「&＋」を導入して逃げた。これにより，



＝「C4&＋C4」＝「C2」



=「C4&D4」

とすればよいことになる。だが，やはり手抜きは手抜きなので，

「C4&+D4」=「C4&+C4」=「C2」

だったりする。少々面倒であるが，「&」と「&+」を使い分けることで我慢していただきたい。

てなわけでリスト11-6のサンプルプログラムを見ていただきたい。基本的な使い方を説明すると，まずは演奏に先立って「TEMPO 0」を実行すること。それから，「MUSIC A$；：MUSIC B$」=「MUSIC A$+B$」である。このセミコロンはなかなかに強力で，これにより1パートが255文字以内というマヌケな制限が消える。なお演奏を止めるには CTRL +D である。

では動作確認である。縁起もんだから馬鹿丁寧にやる。

1) NEW ON &HE000を実行。
2) LOADM "MML.BIN"
3) 「VIP」のディスクから，LOADM "CHOICED VOICE.VTD", &HB000で，音色のデータをロードする（もちろんVIPで作ったデータならほかのものでもかまわない)。
4) CALL &HA8B0を実行。

これでMMLが使えるようになるのである。ちなみに$B000_H$〜$B18F_H$には「音色名」が入っており，**リスト11-7**を実行するとそれの一覧表を見ることができる。

**リスト11-7 音色名の表示**

```
100 FOR I=1 TO 20:J=I+20
110 PRINT #0,I,MEM$(&HB000+(I-1)*10,10),
120 PRINT #0,J,MEM$(&HB000+(J-1)*10,10)
130 NEXT
```

ところで，ちゃんと動くことが分かったら，次にはBASICに組み込んでしまいたいところであるが，面倒になったのでそこまではやってない。私がやったのは，"Start up.Bas"の最後に実行されるNEWをNEW ON &HB800にして，自動的に機械語領域を確保するようにしたことと，

100 LOADM "MML.BIN"
110 LOADM "CHOICED VOICE.VTD",&HB000
120 CALL &HA8B0

という小さなプログラムをディスクに用意しておくようにしたことだけである。こうしておけばBASICが立ち上がった後で，上記のプログラムをRUNするだけでMMLが使えるようになる。

念のために繰り返すが，音色データはLOADM "(ファイル名)",&HB000でロードアドレスを指定するのを忘れないように。さもないとBASICを壊して暴走するのである。

# おまけである

**表 11-11** に VIP の音色データのフォーマットを示す。この章で作った MML では，30 バイト目の KEY TRANSPOSE と 35 バイト目の SYC/LFO DELEY は使われていない。また 31，32，33，34 バイト目のデータは，**最後の I コマンドで指定されたものが有効になる**。よって必要があるならば，実行される I コマンドの順を変えたり，Y コマンドを使うこと。ただ，どうやら LFO に関しては無視してもかまわないようである（自信はないが）。

# 解説するのである

最後に音量の設定についてちょっとだけ説明しておく。FM 音源で音量を変えるには，「アルゴリズム（ALG）を見て，キャリアの TL だけを変える」のである。いままで説明しなかったのだが，オペレータにはキャリアとモジュラの 2 とおりがあるのだ。アルゴリズム 0 〜 3 ではキャリアは OP4 だけ，アルゴリズム 4 では OP4 と OP2 の二つ，アルゴリズム 5，6 では OP4，OP3，OP2 の三つ，アルゴリズム 7 ではすべてがキャリアとなっている。図 11-5 を見れば分かるだろうが，キャリアとはすなわち最終段にある（他のオペレータを変調しない）オペレータのことである。

実際にリスト 11-4 の $AF61_H$ 〜$AFBB_H$ でどんなことをやっているのかというと，

1） 音色データの TL の所をサーチして，最小の TL（つまり最大の音量）になっているキャリアを見つける。

2） 1 で見つけたキャリアの TL を V コマンドで指定された値にするための差を D とする（D＝127－V－TL：TL＝ 0 なら V＝127 であることに注意）。

3） すべてのキャリアの TL に D を加えて，OPM に書き込む。

という具合である。

注意深く見るとあちこちに手抜きが発見できるがあまり**堅いことは言わないでいただきたい。**

表 11-11　VIP の音色データのフォーマット

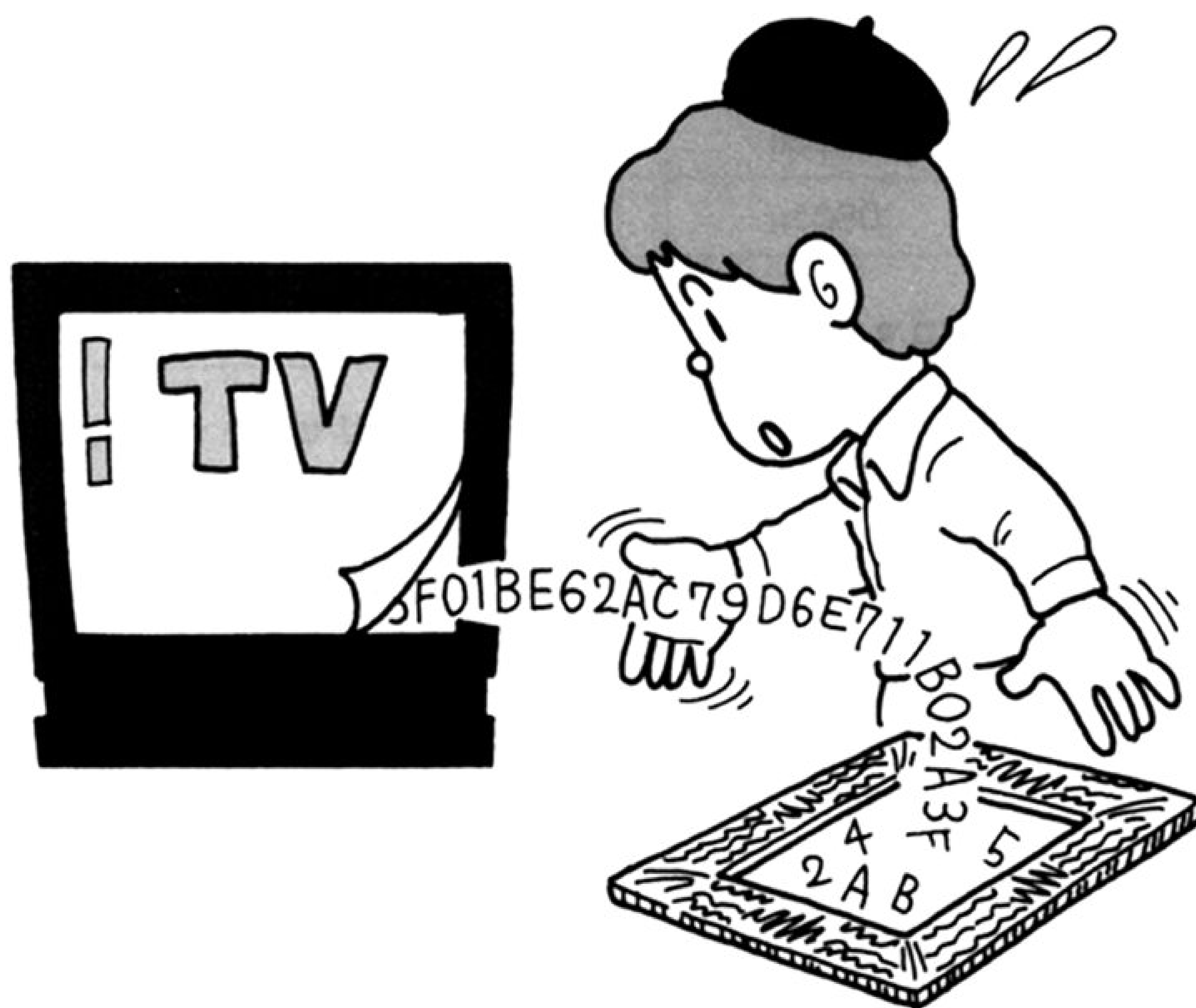
| | |
|---|---|
| 0000H | 音色名　10文字×40 |
| 0190H | パラメータ　36バイト×40 |
| 072FH | |

パラメータの内訳

| No. | OP | bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | | RL | | FB | | | ALG | | |
| 1 | | | PMS | | | | | AMS | |
| 2<br>3<br>4<br>5 | 1<br>2<br>3<br>4 | | Detune1 | | | Phase Multiply | | | |
| 6<br>7<br>8<br>9 | 1<br>2<br>3<br>4 | | Total Level Attenuation | | | | | | |
| 10<br>11<br>12<br>13 | 1<br>2<br>3<br>4 | Key Scaling | | | Attack Rate Speed | | | | |
| 14<br>15<br>16<br>17 | 1<br>2<br>3<br>4 | AMS -EN | | | 1st Decay Rate Speed | | | | |
| 18<br>19<br>20<br>21 | 1<br>2<br>3<br>4 | Detune 2 | | | 2nd Decay Rate Speed | | | | |
| 22<br>23<br>24<br>25 | 1<br>2<br>3<br>4 | 1st Decay Level | | | | Release Rate | | | |
| 26<br>27<br>28<br>29 | 1<br>2<br>3<br>4 | Key Scale Table, | | | | Key Scale Depth | | | |
| 30 | | Key Transpose | | | | | | | |
| 31 | | LFO Frequency | | | | | | | |
| 32 | | 1 | PMD | | | | | | |
| 33 | | 0 | AMD | | | | | | |
| 34 | | | | | | | | W/F | |
| 35 | | SYC | LFO Deley | | | | | | |

# 第12章

# カラーイメージボード




カラーイメージボードで取り込むのである

# 第12章 カラーイメージボードで取り込むのである・・・・・・・・・・・・

この章で取り上げるカラーイメージボードにはCZ-8BV1とCZ-8BV2の二つがあるわけだ。BV1とBV2の主な違いは，BV2では，スクランブル（ハーフトーン）回路のモードが増えたことと，ハイスピードモノクロなどという技が付いたこと，それに320/640ドット，それぞれに対応するスクランブルモードが付いたことなどである。それはともかく，**論より証拠**である。さっさと**表12-1，12-2**を見ていただきたい。

まず表12-1のCZ-8BV1である。注意すべき点は，コマンド$10_H$は1/1画面モード，リセットコマンド，スクランブルONの三つの機能を兼ねているということである。よって，1/4画面および1/16画面のときに，

**スクランブルOFF → ON**

に切り換えるなどするには，$10_H$を出してスクランブルをONにした後で，もう一度$20_H$か$40_H$を出力して，画面モードを設定し直さなければならない。また，スクランブルOFFの状態のままで，画面モードを1/4 → 1/1とするときなどは，1/1画面モードにすると同時にスクランブルがONになってしまうので，次に$80_H$を出力して，スクランブルをOFFにしなければならない。

表12-1　CZ-8BV1

| | | |
|---|---|---|
| $0800_H$ | カラーイメージボードコントロール | OUT |
| $0801_H$ | 画像データ読み込み | IN |

●コマンド

| | |
|---|---|
| $10_H$ 1) | 1/1画面モード＋リセット＋スクランブルON |
| $20_H$ 1) | 1/4画面モード |
| $40_H$ 1) | 1/16画面モード |
| $80_H$ 1) | スクランブル回路OFF |
| $01_H$ | 次の1ラインを読み始める |
| $02_H$ | 次のプレーン(色)を読み始める（B→R→G→B→……） |
| $04_H$ 2) | コンピュータアクセスモード（バッファを読み始める） |
| $08_H$ 2) | ビデオ信号入力モード（ビデオ信号をデジタイズし続ける） |

注1：1）および2）のコマンドを出力した後は，最低16.7msのウエイトを取ること（余裕を持って17ms以上が望ましい）。ただし，1）の後に1）のコマンドが続くときならウエイトは必要ない。

注2：コマンド$01_H$を出力後，2ms以内に1ラインのデータを転送し終わること。また，このコマンド$01_H$を連続して出力するときは，6μs以上の遅延ウエイトを取ること。

注3：1/4，1/16画面モードは横方向だけの縮小であるから，縦方向はソフトウェアで「間引く」必要がある。

次に表12-2がCZ-8BV2である。I/Oアドレスが＋2して，コマンドが四つ増えていることに注目。ここで注意しておかなければならないのが，CZ-8BV2は，リセット時にスク

ランブルが「モード２」になっているということである。すなわち，CZ-8BV2とはちょいと違うスクランブルをするのである。具体的にどう違うかというと，はっきり言ってモード２の方がより自然な中間色を表現しているのである。モード１では斜めの縞がどうしても目立ったわけであるが，モード２ではそういうことは少なくなっているのである。また，モード２のとき（だけ）はさらにスクランブルの仕方で，320/640ドットモードを選択できるようになっている。これは「スクランブルの仕方」なのである。もともとは640ドットモードなのであるが，もしも本体の画面モードが320ドット（WIDTH 40）だった場合は，320ドットモードのスクランブルの方がより自然に見えるようである。

表12-2　CZ-8BV2

| | | |
|---|---|---|
| 0802H | カラーイメージボードコントロール | OUT |
| 0803H | 画像データ読み込み | IN |

●コマンド

| | |
|---|---|
| 10H 1) | 1/1画面モード＋リセット＋スクランブルON |
| 20H 1) | 1/4画面モード |
| 40H 1) | 1/16画面モード |
| 80H 1) | スクランブル回路OFF |
| 01H | 次の1ラインを読み始める |
| 02H | 次のプレーン(色)を読み始める（B→R→G→B→……） |
| 04H 2) | コンピュータアクセスモード（バッファを読み始める） |
| 08H 2) | ビデオ信号入力モード（ビデオ信号をデジタイズし続ける） |
| 81H | スクランブルをモード1（BV1と同じ）にする |
| 83H | 320ドットモードのスクランブル |
| 84H | 640ドットモードのスクランブル |
| 85H | ハイスピードモノクロモード |

注1：CZ-8BV2では，リセット直後のスクランブルモードは「モード2の640ドットモード」である。よってCZ-8BV1と同じ動作をさせるためには，コマンド81Hを実行する必要がある。
注2：320/640ドットモードはモード2のときだけ意味を持つ（モード1では両者は同じ）。

# 実践するのである

さっさとサンプルを出してしまうのである。**リスト12-1，12-2，12-3，12-4**である。リスト12-1はX1用，リスト12-2は機械語部分，リスト12-3はturbo用，リスト12-4は機械語部分である。リスト12-1，12-3ともに，一般的な変更箇所は，

- 130行のI/Oアドレス
- 170〜190行のモード設定

となっている。CZ-8BV1を使っているならば，130行を，

　130 CP＝&H800：DP＝&H801

としていただきたい。170〜190行は画面の大きさの選択である。「’」(REM)で殺してない行が選択されるわけだ。おっと，今気が付いたがこの場合はどちらもWIDTH 80のモードで走らせていただきたい。

リスト 12-1　X1 用画像取り込みプログラム(遅い)

```
100 'FOR X1
110 INIT:CLS4:DEFINT B-Z
120 CLEAR &HE000
130 CP=&H802:DP=&H803
140 MEM$(&HE000,16)=HEXCHR$("EB 4E 23 46 16 50 D9 01 03 08 ED 78 D9 ED 79 03")
150 MEM$(&HE010,15)=HEXCHR$("15 D9 20 F6 C9")
160 DEFUSR0=&HE000
170  XS=80:YS=200:MD=&H10:SL=1  :'640*200
180 'XS=40:YS=96 :MD=&H20:SL=2  :'320*96
190 'XS=20:YS=48 :MD=&H40:SL=4  :'160*48
200 MEM$(&HE008,2)=MKI$(DP)
210 POKE &HE005,XS
220 OUT CP,&H10          :'RESET
230 OUT CP,MD            :'1/1,1/4,1/16 MODE
240 '
250 OUT CP,&H4           :'COMP ACCESS (LATCH)
260 PAUSE 1
270 ADR=&H4000:GOSUB"IS"
280 ADR=&H8000:GOSUB"IS"
290 ADR=&HC000:GOSUB"IS"
300 OUT CP,&H8           :'VIDEO ACCESS
310 PAUSE 1
320 GOTO 250
330 END
340 '
350 LABEL"IS"
360 OUT CP,&H2           :'RGB
370 FOR I=1 TO 24
380   OUT CP,1           :'SET NEXT 1 LINE
390 NEXT
400 FOR Y=1 TO YS
410   D=INP(DP):D=INP(DP):D=INP(DP):'SKIP 3
420   D$=USR0(MKI$(ADR))
430   ADR=ADR+&H800
440   IF (ADR AND &H3800)=0 THEN ADR=ADR-(&H4000-80)
450   FOR I=1 TO SL
460     OUT CP,1:'PAUSE 0                        :'SET NEXT 1 LINE
470   NEXT
480 NEXT
490 RETURN
```

リスト 12-2　カラーイメージボード→G-RAM へ 1 ライン転送

```
                                .Z80
                                .PHASE  0E000H  ;OR ANY PLACE
                        ;
E000    EB                      EX      DE,HL
E001    4E                      LD      C,(HL)
E002    23                      INC     HL
E003    46                      LD      B,(HL)  ;BC=VRAM ADDR.
E004    16 50                   LD      D,80    ;XS
E006    D9                      EXX
E007    01 0803                 LD      BC,0803H        ;DPORT
E00A    ED 78           LOOP:   IN      A,(C)   ;GET 1 BYTE
E00C    D9                      EXX
E00D    ED 79                   OUT     (C),A   ;TO VRAM
E00F    03                      INC     BC      ;INC VRAM ADDR.
E010    15                      DEC     D       ;DEC COUNTER
E011    D9                      EXX
E012    20 F6                   JR      NZ,LOOP
                        ;
E014    C9                      RET
                        ;
                                END
```

リスト 12-3　turbo 用画像取り込みプログラム(遅い)

```
100 'FOR turbo
110 INIT:CLS4:DEFINT B-Z
120 CLEAR &HE000
```

```
130 CP=&H802:DP=&H803
140 MEM$(&HE000,12)=HEXCHR$("78 EB 01 80 1F 04 ED A3 3D 20 FA C9")
150 '
160 DEFUSR0=&HE000
170  XS=80:YS=200:MD=&H10:SL=1   :'640*200
180 'XS=40:YS=96 :MD=&H20:SL=2   :'320*96
190 'XS=20:YS=48 :MD=&H40:SL=4   :'160*48
200 OUT CP,&H10           :'RESET
210 OUT CP,MD             :'1/1,1/4,1/16 MODE
220 DMA$=DMA$+CHR$(&B1111001)+MKI$(&H4000)+MKI$(XS-1)
230 DMA$=DMA$+CHR$(&B11100,&B101000,&B11001101)
240 DMA$=DMA$+MKI$(DP)+CHR$(&B10011010)
250 D$=USR0(DMA$)
260 PAUSE 1
270 '
280 DMA$=CHR$(&B11001)+MKI$(0)+HEXCHR$("CF 87")
290 OUT CP,&H4            :'COMP ACCESS (LATCH)
300 PAUSE 1
310 ADR=&H4000:GOSUB"IS"
320 ADR=&H8000:GOSUB"IS"
330 ADR=&HC000:GOSUB"IS"
340 OUT CP,&H8            :'VIDEO ACCESS
350 PAUSE 1
360 GOTO 290
370 END
380 '
390 LABEL"IS"
400 OUT CP,&H2            :'RGB
410 FOR I=1 TO 24
420   OUT CP,1            :'SET NEXT 1 LINE
430 NEXT
440 FOR Y=1 TO YS
450   D=INP(DP):D=INP(DP):D=INP(DP):'SKIP 3
460   MID$(DMA$,2,2)=MKI$(ADR)
470   D$=USR0(DMA$)
480   ADR=ADR+&H800
490   IF (ADR AND &H3800)=0 THEN ADR=ADR-(&H4000-80)
500   FOR I=1 TO SL
510     OUT CP,1:'PAUSE 0                        :'SET NEXT 1 LINE
520   NEXT
530 NEXT
540 RETURN
```

**リスト 12-4　文字列→DMA 転送**

```
                                .Z80
                                .PHASE  0E000H  ;OR ANY PLACE
                        ;
                        ;DE=ADDRESS TO DATA,B=COUNTER
E000    78                      LD      A,B
E001    EB                      EX      DE,HL           ;HL POINTS DATA
E002    01 1F80                 LD      BC,1F80H        ;DMA I/O ADDR.
E005    04              LOOP:   INC     B
E006    ED A3                   OUTI
E008    3D                      DEC     A
E009    20 FA                   JR      NZ,LOOP
E00B    C9                      RET
                        ;
                                END
```

プログラムの説明に入る。まずはリスト 12-1 である。大事なのは 140，150 行で機械語ルーチンを用意している点である。この機械語ルーチンは手抜きしてあるので，200，210 行で一部を書き換えたりしていてあまりよくない。

さて，まずは 220 行である。これはコントロールポートに $10_H$ を出力しているわけだ。これが何かと見るならば，表 12-1 より明らかなように，「1/1 画面モード＋リセット＋スク

ランプル ON」なわけだ。その次の 230 行では再びコントロールポートに MD を出力している。これは何かと見るならば，結局 MD は $10_H$，$20_H$，$40_H$のうちのどれかなのである。MD＝$10_H$の場合は，先程と同じ命令を繰り返すことになるわけだから，ま，無駄ということになる。そこで MD＝$20_H$の場合を説明するのである。

コントロールポートに $20_H$が出力される前は，カラーイメージボードは 1/1 画面モードで**シャコシャコと**ビデオ入力をデジタイズしていたわけである。すなわち，カラーイメージボードはビデオ画面を 640 ドット×200 ドット＝80 バイト×200 ライン（実を言うと本当はもっと大きいのだが)のデータに変換していたのである。そこへ $20_H$というコマンドが来たわけである。そうすると，カラーイメージボードはそれまでの 640×200 をやめて，ビデオ信号を 320 ドット×200 ドット＝40 バイト×200 ラインのデータに変換するようになるのだ。縦の方は 200 ラインのまま変わらないということに注意。もしもこれが $20_H$でなく $40_H$であったなら，160 ドット×200 ドット＝20 バイト×200 ラインとなる（ここでは簡単に説明するために「200 ライン」と書いてあるが，実はカラーイメージボードはもっと多くのラインを取り込んでいるのである。実際に表示するのはそのうちの 200 ラインということになる)。

これらのデータを変に潰れたり，伸びたりせずに表示させるには，コンピュータの画面モード（WIDTH）との関係で**表 12-3** のようにすることになる。

**表 12-3 カラーイメージボードと本体の画面モード**

| | WIDTH 40 | WIDTH 80 |
|---|---|---|
| 1/1＝$10_H$ | | 80バイト×200ラインをそのままG-RAMへ（画面全体に表示） |
| 1/4＝$20_H$ | 40バイト×200ラインをそのままG-RAMへ（画面全体に表示） | 40バイト×200ラインを1ラインおきにG-RAMへ（画面の1/4に表示） |
| 1/16＝$30_H$ | 20バイト×200ラインを1ラインおきにG-RAMへ（画面の1/4に表示） | 20バイト×200ラインを3ラインおきにG-RAMへ（画面の1/16に表示） |

さてここからが佳境である。リスト 12-1 の 250 行でコントロールポートに $04_H$を出力している。これは表 12-1 にもあるように，「コンピュータアクセスモード」なのである。早い話が，カラーイメージボードはこのコマンドを受け取ると，それまでシャカシャカとデジタイズしていたのを**やめて，**「よっしゃよっしゃ，デジタイズしたデータが欲しいんだな。ようし，イキのいいやつをデジタイズして渡してやらあ。受け取りな，ほうら」となるわけである。すなわち，カラーイメージボードは，自分の持っているバッファを一杯にした後で「データ転送モード」に入るのである（表 12-1 の注 1 にもあるように，バッファが一杯になるまで，最悪の場合 17 ms 待たなければならない)。

その次の 270〜290 行では青，赤，緑の順でカラーイメージボードから G-RAM に転送している。

というわけで，350 行からのサブルーチン〝IS〞の説明である。

まずは 360 行で「2」を OUT している。これは言わずと知れた B→R→G→B……な

わけだ。次に370〜390行のループであるが，これは24行の空読みである。先程書いたように，カラーイメージボードは実際は200ラインよりも多くのライン数を取り込んでいるのだ。それで，ちょうどおいしい所だけを表示するために，最初の方は空読みして捨てているのである。それから410行である。今度は3バイトの空読みである。何でこうするのかというと，それはカラーイメージボードのハードがそうなっているからである。試しに410行を削除してみると，表示が少し右にずれるのが分かるだろう。後は420行で1ライン転送し，430行，440行でG-RAMアドレスを1ライン下げ，450〜470行で（カラーイメージボード側の）ラインのスキップをやっている。

以上で一とおり説明は終わりである。170〜190行で注釈にしてある行を復活して大きさの違う画面を表示したり，YSの値を大きくしたり，SLの値をいじったりして楽しんでいただきたい。

リスト12-3のturbo版は，データの転送にDMAを使っていること以外は変わったことはないので，説明は省略する。

## 気分は近未来である

さて，ここからぼちぼちフィニッシュに持ち込むのである。そして一体何をやるかというと，「BASIC＋動画面のウィンドウ」なのである。

種を明かすと，CTCの割り込みを使って，BASICと画像取り込みが並列動作してしまうのである。言葉で言うと簡単に聞こえてしまうが，これはちょっと他機種ではまねができないワザであろう。BASICがちゃんと動いている同じ画面で，1/16（もしくは1/4）の画面がリアルタイムで動いているのである。これを見てむむむとうならない奴はいないであろう。

で，このよーにBASIC＋画像取り込みが（一応）マルチタスクすると，どういうメリットがあるかというと，

1) 他機種のユーザーに自慢できる
2) 他機種のユーザーが驚く
3) 他機種のユーザーが落ち込む
4) 他機種のユーザーが寝返る
5) テレビを見ながらプログラムや，機械語のダンプリストを入力できる（スーパーインポーズよりずっと目触りがよい）
6) 近未来の気分を味わえる

となっている。とにもかくにも，8色ながらも「動画面のウィンドウ」なのである。こんなSFっぽいことを，8ビットのturbo（とCTCを持つX1）でできてしまうのだだだっ！と逆上しつつ，プログラムの説明になだれ込むのであった。

**リスト12-5**がX1用のアセンブルリスト，**リスト12-6**がそのダンプリスト（CRCチェックサム付き：付録Aを参照），**リスト12-7**がturbo用のアセンブルリスト，**リスト12-8**がそのCRCチェックサム付きダンプリストとなっている。

そして使い方は**リスト 12-9～12-12** である。

順に説明しよう。

おのおののダンプリストを打ち込み，リスト 12-6 は，

SAVEM "BV.CPU", &HEF00, &HEFC6

でセーブする。

リスト 12-8 ならば，

SAVEM "BV.DMA", &HEF00, &HEFEE

である。チェックサム（およびCRC）の確認を確実にすること。

**リスト 12-5　X1 用画像取り込みプログラム**

```
                                .Z80
                                .PHASE  0EF00H
                        ;
                        ;
                        ;COMMANDS       10H     RESET,1/1
                        ;               20H     1/4
                        ;               40H     1/16
                        ;               80H     HALF TONE OFF
                        ;
                        ;               01H     NEXT 1 LINE
                        ;               02H     NEXT PLANE(B->R->G->B..)
                        ;               04H     COMP ACCESS
                        ;               08H     VIDEO ACCESS
                        ;
                        ;CZ-8BV2
                        ;               81H     HALF TONE=MODE1
                        ;               83H     X=320 MODE
                        ;               84H     X=640 MODE
                        ;               85H     MONO COLOR MODE
                        ;
EF00    50              XS:     DB      80      ;X SIZE
EF01    C8              YS:     DB      200     ;Y SIZE
EF02    01              UPPASS: DB      1       ;UP PASS LINES
EF03    01              LTPASS: DB      1       ;LEFT PASS BYTES
                        ;
EF04    01              STIME:  DB      1       ;SKIP LINES
                        ;
EF05    10 00           MODES:  DB      10H,00H
EF07    00 00 00 00             DB      00H,00H,00H,00H,00H
EF0B    00
                        ;
EF0C    0802            CPORTA: DW      0802H   ;OR 800H
EF0E    4000            TOPA:   DW      4000H
                        ;
                        ;=========================================
EF10    CD EFAC         START:  CALL    INIT    ;INIT BV
EF13    F3              ENT:    DI
EF14    F5                      PUSH    AF
EF15    C5                      PUSH    BC
EF16    D5                      PUSH    DE
EF17    E5                      PUSH    HL
EF18    D9                      EXX
EF19    C5                      PUSH    BC
EF1A    D5                      PUSH    DE
EF1B    E5                      PUSH    HL      ;SAVE REGISTERS
                        ;
EF1C    ED 4B EF0C              LD      BC,(CPORTA)
EF20    3E 04                   LD      A,04H   ;COMP ACCESS
EF22    ED 79                   OUT     (C),A   ;LATCH
EF24    D9                      EXX             ;BC'=(CPORTA)
EF25    CD EFBE                 CALL    DELAY
                        ;
EF28    2A EF0E                 LD      HL,(TOPA)       ;VRAM ADDRESS
EF2B    E5                      PUSH    HL
EF2C    CD EF52                 CALL    IMAGES  ;BLUE
                        ;
EF2F    E1                      POP     HL
EF30    E5                      PUSH    HL
EF31    CB B4                   RES     6,H
EF33    CB FC                   SET     7,H
EF35    CD EF52                 CALL    IMAGES  ;RED
```

```
EF38    E1          ;
EF39    CB FC              POP     HL
EF3B    CD EF52            SET     7,H
                           CALL    IMAGES  ;GREEN
                    ;
EF3E    3E 08              LD      A,08H   ;VIDEO ACCESS
EF40    D9                 EXX             ;BC'=(CPORTA)
EF41    ED 79              OUT     (C),A
EF43    D9                 EXX
                    ;
EF44    CD EFBE            CALL    DELAY
                    ;
EF47    E1                 POP     HL
EF48    D1                 POP     DE
EF49    C1                 POP     BC
EF4A    D9                 EXX
EF4B    E1                 POP     HL
EF4C    D1                 POP     DE
EF4D    C1                 POP     BC
EF4E    F1                 POP     AF      ;LOAD REGISTERS
                    ;
EF4F    FB                 EI
EF50    C9                 RET             ;OR RETI
EF51    00                 NOP
                    ;======================================
                    ;
EF52    3E 02       IMAGES: LD     A,02H   ;NEXT PLANE
EF54    D9                 EXX             ;BC'=(CPORTA)
EF55    ED 79              OUT     (C),A
EF57    D9                 EXX
                    ;
EF58    CD EF91            CALL    DUMYV   ;SKIP LINES
EF5B    3A EF01            LD      A,(YS)  ;LINE COUNT
EF5E    5F                 LD      E,A     ;E=COUNTER
EF5F    CD EF9F     IMAY:  CALL    DUMYH   ;SKIP BYTES
EF62    44                 LD      B,H
EF63    4D                 LD      C,L     ;COPY ADDRESS
EF64    3A EF00            LD      A,(XS)
EF67    57                 LD      D,A     ;COUNTER
EF68    D9                 EXX             ;BC'=(CPORTA)
EF69    03                 INC     BC      ;BC'=(DPORTA)
EF6A    ED 78       POO:   IN      A,(C)   ;READ 1 BYTE
EF6C    D9                 EXX
EF6D    ED 79              OUT     (C),A   ;TO VRAM
EF6F    03                 INC     BC      ;INC VRAM ADDR.
EF70    15                 DEC     D       ;DEC COUNTER
EF71    D9                 EXX
EF72    C2 EF6A            JP      NZ,POO
EF75    0B                 DEC     BC      ;BC'=(CPORTA)
EF76    D9                 EXX
                    ;
EF77    3A EF04            LD      A,(STIME)
EF7A    CD EF94            CALL    NEXTL   ;SKIP LINES
                    ;
EF7D    3E 08              LD      A,8     ;DOWN 1 LINE
EF7F    84                 ADD     A,H
EF80    67                 LD      H,A
EF81    E6 38              AND     38H
EF83    C2 EF8C            JP      NZ,IMANEX
EF86    01 3FB0            LD      BC,4000H-80
EF89    B7                 OR      A
EF8A    ED 42              SBC     HL,BC
EF8C    1D          IMANEX: DEC    E
EF8D    C2 EF5F            JP      NZ,IMAY ;LOOP
EF90    C9                 RET
                    ;
                    ;?? LINES PASS
EF91    3A EF02     DUMYV: LD      A,(UPPASS)
                    ;
EF94    D9          NEXTL: EXX             ;BC'=(CPORTA)
EF95    16 01              LD      D,01H   ;1L DOWN
EF97    ED 51       NEXTL1: OUT    (C),D
EF99    3D                 DEC     A
EF9A    C2 EF97            JP      NZ,NEXTL1
EF9D    D9                 EXX
EF9E    C9                 RET
                    ;
                    ;?? BYTES SKIP
EF9F    D9          DUMYH: EXX             ;BC'=(CPORTA)
EFA0    0C                 INC     C       ;BC'=(DPORTA)
EFA1    3A EF03            LD      A,(LTPASS)
                    ;
```

```
EFA4     ED 50            DUMYHL: IN      D,(C)
EFA6     3D                       DEC     A
EFA7     20 FB                    JR      NZ,DUMYHL
EFA9     0D                       DEC     C
EFAA     D9                       EXX             ;BC'=(CPORTA)
EFAB     C9                       RET
                          ;
EFAC     ED 4B EF0C       INIT:   LD      BC,(CPORTA)
EFB0     21 EF05                  LD      HL,MODES
EFB3     7E               INIT0:  LD      A,(HL)
EFB4     23                       INC     HL
EFB5     B7                       OR      A
EFB6     C8                       RET     Z       ;END MODE SET
EFB7     ED 79                    OUT     (C),A   ;SET BV MODE
EFB9     CD EFBE                  CALL    DELAY
EFBC     18 F5                    JR      INIT0
                          ;
EFBE     11 0A37          DELAY:  LD      DE,2615 ;10
EFC1     1B               DELAYL: DEC     DE      ;6
EFC2     7A                       LD      A,D     ;4
EFC3     B3                       OR      E       ;4
EFC4     20 FB                    JR      NZ,DELAYL       ;12/7
EFC6     C9                       RET             ;10
                          ;10+10+(6+4+4+12)*2615-12+7=68005
                          ;                       =17.00125ms
                          ;
                                  END
```

リスト 12-6「リスト 12-5」のダンプリスト　〝BV. CPU〞でセーブ

```
EF00 50 C8 01 01 01 10 00 00 : 2B
EF08 00 00 00 00 02 08 00 40 : 4A
EF10 CD AC EF F3 F5 C5 D5 E5 : CF
EF18 D9 C5 D5 E5 ED 4B 0C EF : 8B
EF20 3E 04 ED 79 D9 CD BE EF : FB
EF28 2A 0E EF E5 CD 52 EF E1 : FB
EF30 E5 CB B4 CB FC CD 52 EF : 39
EF38 E1 CB FC CD 52 EF 3E 08 : FC
EF40 D9 ED 79 D9 CD BE EF E1 : 73
EF48 D1 C1 D9 E1 D1 C1 F1 FB : CA
EF50 C9 00 3E 02 D9 ED 79 D9 : 21
EF58 CD 91 EF 3A 01 EF 5F CD : A3
EF60 9F EF 44 4D 3A 00 EF 57 : 9F
EF68 D9 03 ED 78 D9 ED 79 03 : 83
EF70 15 D9 C2 6A EF 0B D9 3A : 27
EF78 04 EF CD 94 EF 3E 08 84 : 0D
---------------------------------
SUM: F5 DA 90 88 42 94 1F 75 92ED

EF80 67 E6 38 C2 8C EF 01 B0 : 73
EF88 3F B7 ED 42 1D C2 5F EF : 52
EF90 C9 3A 02 EF D9 16 01 ED : D1
EF98 51 3D C2 97 EF D9 C9 D9 : 51
EFA0 0C 3A 03 EF ED 50 3D 20 : D2
EFA8 FB 0D D9 C9 ED 4B 0C EF : DD
EFB0 21 05 EF 7E 23 B7 C8 ED : 22
EFB8 79 CD BE EF 18 F5 11 37 : 48
EFC0 0A 1B 7A B3 20 FB C9    : 36
---------------------------------
SUM: 6B 48 EC 62 A6 E2 15 98 168C
```

リスト 12-7　X1 turbo 用画像取り込みプログラム

```
                                  .Z80
                                  .PHASE  0EF00H
                          ;
1F80                      DMA     EQU     1F80H
                          ;
                          ;COMMANDS       10H     RESET,1/1
                          ;               20H     1/4
                          ;               40H     1/16
                          ;               80H     HALF TONE OFF
                          ;
                          ;               01H     NEXT 1 LINE
                          ;               02H     NEXT PLANE(B->R->G->B..)
                          ;               04H     COMP ACCESS
                          ;               08H     VIDEO ACCESS
                          ;
                          ;CZ-8BV2
                          ;               81H     HALF TONE=MODE1
                          ;               83H     X=320 MODE
                          ;               84H     X=640 MODE
                          ;               85H     MONO COLOR MODE
                          ;
EF00     50               XS:     DB      80      ;X SIZE
EF01     C8               YS:     DB      200     ;Y SIZE
EF02     01               UPPASS: DB      1       ;UP PASS LINES
EF03     01               LTPASS: DB      1       ;LEFT PASS BYTES
                          ;
```

```
EF04    01                  STIME:  DB      1           ;SKIP LINES
                            ;
EF05    10 00               MODES:  DB      10H,00H
EF07    00 00 00 00                 DB      00H,00H,00H,00H,00H
EF0B    00
                            ;
EF0C    0802                CPORTA: DW      0802H       ;OR 800H
EF0E    4000                TOPA:   DW      4000H
                            ;
                            ;=======================================
EF10    CD EFAA             START:  CALL    INIT        ;INIT BV,DMA
EF13    F3                  ENT:    DI
EF14    F5                          PUSH    AF
EF15    C5                          PUSH    BC
EF16    D5                          PUSH    DE
EF17    E5                          PUSH    HL
EF18    D9                          EXX
EF19    C5                          PUSH    BC
EF1A    D5                          PUSH    DE
EF1B    E5                          PUSH    HL          ;SAVE REGISTERS
                            ;
EF1C    ED 4B EF0C                  LD      BC,(CPORTA)
EF20    3E 04                       LD      A,04H       ;COMP ACCESS
EF22    ED 79                       OUT     (C),A       ;LATCH
EF24    D9                          EXX                 ;BC'=(CPORTA)
EF25    CD EFE6                     CALL    DELAY
                            ;
EF28    2A EF0E                     LD      HL,(TOPA)           ;VRAM ADDRESS
EF2B    E5                          PUSH    HL
EF2C    CD EF52                     CALL    IMAGES      ;BLUE
                            ;
EF2F    E1                          POP     HL
EF30    E5                          PUSH    HL
EF31    CB B4                       RES     6,H
EF33    CB FC                       SET     7,H
EF35    CD EF52                     CALL    IMAGES      ;RED
                            ;
EF38    E1                          POP     HL
EF39    CB FC                       SET     7,H
EF3B    CD EF52                     CALL    IMAGES      ;GREEN
                            ;
EF3E    3E 08                       LD      A,08H       ;VIDEO ACCESS
EF40    D9                          EXX                 ;BC'=(CPORTA)
EF41    ED 79                       OUT     (C),A
EF43    D9                          EXX
                            ;
EF44    CD EFE6                     CALL    DELAY
                            ;
EF47    E1                          POP     HL
EF48    D1                          POP     DE
EF49    C1                          POP     BC
EF4A    D9                          EXX
EF4B    E1                          POP     HL
EF4C    D1                          POP     DE
EF4D    C1                          POP     BC
EF4E    F1                          POP     AF          ;LOAD REGISTERS
                            ;
EF4F    FB                          EI
EF50    C9                          RET                 ;OR RETI
EF51    00                          NOP
                            ;=======================================
                            ;
EF52    3E 02               IMAGES: LD      A,02H       ;NEXT PLANE
EF54    D9                          EXX                 ;BC'=(CPORTA)
EF55    ED 79                       OUT     (C),A
EF57    D9                          EXX
                            ;
EF58    CD EF8F                     CALL    DUMYV       ;SKIP LINES
EF5B    3A EF01                     LD      A,(YS)      ;LINE COUNT
EF5E    5F                          LD      E,A         ;E=COUNTER
EF5F    CD EF9D             IMAY:   CALL    DUMYH       ;SKIP BYTES
EF62    01 1F80                     LD      BC,DMA      ;DMA ADDRESS
EF65    3E 19                       LD      A,00011001B       ;WR0
EF67    ED 79                       OUT     (C),A
EF69    ED 69                       OUT     (C),L
EF6B    ED 61                       OUT     (C),H       ;SET VRAM ADDRESS
EF6D    3E CF                       LD      A,0CFH      ;WR6 LOAD
EF6F    ED 79                       OUT     (C),A
EF71    3E 87                       LD      A,87H       ;WR6 ENABLE
EF73    ED 79                       OUT     (C),A
                            ;
```

```
EF75     3A EF04               LD      A,(STIME)
EF78     CD EF92               CALL    NEXTL    ;SKIP LINES
                       ;
EF7B     3E 08                 LD      A,8      ;DOWN 1 LINE
EF7D     84                    ADD     A,H
EF7E     67                    LD      H,A
EF7F     E6 38                 AND     38H
EF81     C2 EF8A               JP      NZ,IMANEX
EF84     01 3FB0               LD      BC,4000H-80
EF87     B7                    OR      A
EF88     ED 42                 SBC     HL,BC
EF8A     1D            IMANEX: DEC     E
EF8B     C2 EF5F               JP      NZ,IMAY ;LOOP
EF8E     C9                    RET
                       ;
                       ;?? LINES PASS
EF8F     3A EF02       DUMYV:  LD      A,(UPPASS)
                       ;
EF92     D9            NEXTL:  EXX              ;BC'=(CPORTA)
EF93     16 01                 LD      D,01H    ;1L DOWN
EF95     ED 51         NEXTL1: OUT     (C),D
EF97     3D                    DEC     A
EF98     C2 EF95               JP      NZ,NEXTL1
EF9B     D9                    EXX
EF9C     C9                    RET
                       ;
                       ;?? BYTES SKIP
EF9D     D9            DUMYH:  EXX              ;BC'=(CPORTA)
EF9E     0C                    INC     C        ;BC'=(DPORTA)
EF9F     3A EF03               LD      A,(LTPASS)
                       ;
EFA2     ED 50         DUMYHL: IN      D,(C)
EFA4     3D                    DEC     A
EFA5     20 FB                 JR      NZ,DUMYHL
EFA7     0D                    DEC     C
EFA8     D9                    EXX              ;BC'=(CPORTA)
EFA9     C9                    RET
                       ;
EFAA     ED 4B EF0C    INIT:   LD      BC,(CPORTA)
EFAE     21 EF05               LD      HL,MODES
EFB1     7E            INIT0:  LD      A,(HL)
EFB2     23                    INC     HL
EFB3     B7                    OR      A
EFB4     28 07                 JR      Z,INIT1 ;END MODE SET
EFB6     ED 79                 OUT     (C),A    ;SET BV MODE
EFB8     CD EFE6               CALL    DELAY
EFBB     18 F4                 JR      INIT0
                       ;
EFBD     3A EF00       INIT1:  LD      A,(XS)
EFC0     6F                    LD      L,A
EFC1     26 00                 LD      H,0
EFC3     2B                    DEC     HL
EFC4     22 EFDE               LD      (DMAXS),HL        ;SET X SIZE
EFC7     2A EF0C               LD      HL,(CPORTA)
EFCA     23                    INC     HL                ;HL=DPORTA
EFCB     22 EFE3               LD      (DMAPDA),HL
                       ;
EFCE     21 EFDD               LD      HL,DMADT
EFD1     3E 09                 LD      A,DMADTE-DMADT
EFD3     01 1F80       SETDMA: LD      BC,DMA
EFD6     04            SDMAL:  INC     B
EFD7     ED A3                 OUTI             ;OUT TO DMA
EFD9     3D                    DEC     A
EFDA     20 FA                 JR      NZ,SDMAL
EFDC     C9                    RET
                       ;
EFDD     61            DMADT:  DEFB    01100001B         ;WR0
EFDE                   DMAXS:  DEFS    2                 ;LENGTH
EFE0     1C                    DEFB    00011100B         ;WR1
EFE1     28                    DEFB    00101000B         ;WR2
EFE2     CD                    DEFB    1100110LB         ;WR4
EFE3                   DMAPDA: DEFS    2                 ;PORT B ADR.
EFE5     9A                    DEFB    10011010B         ;WR5
EFE6                   DMADTE:
                       ;
EFE6     11 0A37       DELAY:  LD      DE,2615 ;10
EFE9     1B            DELAYL: DEC     DE       ;6
EFEA     7A                    LD      A,D      ;4
EFEB     B3                    OR      E        ;4
EFEC     20 FB                 JR      NZ,DELAYL         ;12/7
EFEE     C9                    RET              ;10
```

```
                    ;10+10+(6+4+4+12)*2615-12+7=68005
                    ;                          =17.00125ms
                    ;
                            END
```

リスト 12-8 「リスト 12-7」のダンプリスト〝BV. DMA″でセーブ

```
EF00 50 C8 01 01 01 10 00 00 : 2B
EF08 00 00 00 00 02 08 00 40 : 4A
EF10 CD AA EF F3 F5 C5 D5 E5 : CD
EF18 D9 C5 D5 E5 ED 4B 0C EF : 8B
EF20 3E 04 ED 79 D9 CD E6 EF : 23
EF28 2A 0E EF E5 CD 52 EF E1 : FB
EF30 E5 CB B4 CB FC CD 52 EF : 39
EF38 E1 CB FC CD 52 EF 3E 08 : FC
EF40 D9 ED 79 D9 CD E6 EF E1 : 9B
EF48 D1 C1 D9 E1 D1 C1 F1 FB : CA
EF50 C9 00 3E 02 D9 ED 79 D9 : 21
EF58 CD 8F EF 3A 01 EF 5F CD : A1
EF60 9D EF 01 80 1F 3E 19 ED : 70
EF68 79 ED 69 ED 61 3E CF ED : 17
EF70 79 3E 87 ED 79 3A 04 EF : D1
EF78 CD 92 EF 3E 08 84 67 E6 : 65
---------------------------------
SUM: C0 C8 B0 5D 52 C0 51 0C 3533
```

```
EF80 38 C2 8A EF 01 B0 3F B7 : 1A
EF88 ED 42 1D C2 5F EF C9 3A : 5F
EF90 02 EF D9 16 01 ED 51 3D : 5C
EF98 C2 95 EF D9 C9 D9 0C 3A : 07
EFA0 03 EF ED 50 3D 20 FB 0D : 94
EFA8 D9 C9 ED 4B 0C EF 21 05 : FB
EFB0 EF 7E 23 B7 28 07 ED 79 : DC
EFB8 CD E6 EF 18 F4 3A 00 EF : D7
EFC0 6F 26 00 2B 22 DE EF 2A : D9
EFC8 0C EF 23 22 E3 EF 21 DD : 10
EFD0 EF 3E 09 01 80 1F 04 ED : C7
EFD8 A3 3D 20 FA C9 61 ED 1E : 2F
EFE0 1C 28 CD 0C CD 9A 11 37 : CC
EFE8 0A 1B 7A B3 20 FB C9    : 36
---------------------------------
SUM: B4 77 EE 11 CA 97 49 2B F2C9
```

次に小手しらべである。まず X1 ユーザーはリスト 12-9 を，turbo ユーザーはリスト 12-10 を打ち込む（120 行が 1 行違うだけである）。暴走すると元も子もなくなるのでセーブを忘れないように。

それから RUN すると（あらかじめ機械語部分を LOADM でロードしておいた方がよい），全画面取り込みが始まるのである。それがうまくいったらこっちのものである。[SHIFT] + [BREAK] で止めていただきたい。次に 130〜160 行をリスト 12-11, 12-12 のように打ち換えてそれぞれセーブしていただきたい。おのおのが 1/4 画面，1/16 画面の取り込みを実行するのである。

ここらで CZ-8BV2 を持っている人のための腕慣らしである。リスト 12-11 の 150 行にあるコマンドの並びを，

HEXCHR$(〝10 20 83……

としていただきたい。よく観察した後で，BREAK して，次には，

HEXCHR$(〝10 20 84……

としていただきたい。どうだ，微妙にスクランブルのパターンが違うであろう。これが 320 ドットと 640 ドットモードの違いである。

次に同じようなことを，

HEXCHR$(〝10 20 81……

と，

HEXCHR$(〝10 20 00……

でもやっていただきたい。これがモード 1 とモード 2 の違いである。確かにモード 2 の方が自然な感じであろう。また，モード 1 では，320/640 ドットの違いがないということもチェックしていただきたい。さらには申し訳ないのだが，ハイスピードモノクロに関しては自習ということにする。機械語プログラムを適当に書き換えるなどしてみて欲しい。モノ

クロであるから，パレットを替えて，１画面だけ転送すればよいのである。

**リスト 12-9　X1 用全画面面取り込み**

```
100 INIT          :'1/1 MODE
110 CLEAR &HEF00
120 IF MEM$(&HEF10,3)<>HEXCHR$("CD AC EF") THEN LOADM "BV.CPU"
130 MEM$(&HEF00,4)=CHR$(80,200,24,3)
140 MEM$(&HEF04,1)=CHR$(1)
150 MEM$(&HEF05,7)=HEXCHR$("10 00 00 00 00 00 00")
160 MEM$(&HEF0C,4)=MKI$(&H802)+MKI$(&H4000)
170 CALL &HEF10
180 CALL &HEF13:GOTO 180
```

**リスト 12-10　X1 turbo 用全画面面取り込み**

```
100 INIT          :'1/1 MODE
110 CLEAR &HEF00
120 IF MEM$(&HEF10,3)<>HEXCHR$("CD AA EF") THEN LOADM "BV.DMA"
130 MEM$(&HEF00,4)=CHR$(80,200,24,3)
140 MEM$(&HEF04,1)=CHR$(1)
150 MEM$(&HEF05,7)=HEXCHR$("10 00 00 00 00 00 00")
160 MEM$(&HEF0C,4)=MKI$(&H802)+MKI$(&H4000)
170 CALL &HEF10
180 CALL &HEF13:GOTO 180
```

**リスト 12-11　4 分の 1 画面取り込み用変更点**

```
130 MEM$(&HEF00,4)=CHR$(43,116,1,2)
140 MEM$(&HEF04,1)=CHR$(2)
150 MEM$(&HEF05,7)=HEXCHR$("10 20 00 00 00 00 00")
160 MEM$(&HEF0C,4)=MKI$(&H802)+MKI$(&H4000+37+80*10)
```

**リスト 12-12　16 分の 1 画面取り込み用変更点**

```
130 MEM$(&HEF00,4)=CHR$(21,58,1,2)
140 MEM$(&HEF04,1)=CHR$(4)
150 MEM$(&HEF05,7)=HEXCHR$("10 40 00 00 00 00 00")
160 MEM$(&HEF0C,4)=MKI$(&H802)+MKI$(&H4500+59+80)
```

さて，ここからがキメである。1/16 画面の取り込みプログラム（1/1 画面でも 1/4 画面でもいいが）の 180 行以降を**リスト 12-13**（turbo 用）もしくは**リスト 12-14**（X1 用）のように書き換えるのである。そうすれば BASIC は「OK」とメッセージを出すが，画像取り込みは相変わらず行なわれている。すなわちマルチタスク（？）のでき上がりである。ただし，**X1 用の方は，FM 音源ボード（CZ-8BS1）か，立体視ボード（CZ-8RB1）か，RS-232C・マウスボード（CZ-8BM2）が必要である。**そして，それぞれの場合において 200～220 行にあるように「CTC＝～」のアドレスを書き換えること。リスト 12-14 では FM 音源ボードに対応するようになっている。

このプログラムはなかなかに趣があるわけだが，欠点は少々キーの反応が悪くなることである。turbo の場合は TEMPO 文で適当に調節していただきたい（「TEMPO 30」で画

像取り込みが一番遅くなる)。X1の方では，300行と320行でCTCに送っている値(30と125)を大きくすると画像取り込みが遅くなり，小さくすると速くなる。これについては第5章を参照のこと。ところでリスト12-14では190行で「POKE &H61AE, 1」とやっているが，これはBASICを書き換えて，「LIST」命令を実行してもグラフィックがOFFにならないようにしているのである(ただし「FILES」などではOFFになる)。ちなみにturboでは（リスト12-13ではやってないが)，「POKE &H14F9, 1」とすることによって，「LIST」を実行してもグラフィックがOFFにならなくなる。

なお注意しておくが，リスト12-13の200行は「割り込みを禁止せずにベクトルテーブルを書き換えている」ので，とんでもない反則である。やりすぎると暴走する。また，リスト12-13，12-14は「何度もRUNさせてはいけない」のである。つまり，割り込みを禁止せずに割り込み処理ルーチンのワークをいじることになるからである。すなわち，2度目のRUN以降は**ロシアンルーレット**みたいなものである。

また，プログラム中でDELAYというサブルーチンを使っているが，その時間はドブに捨ててしまっているのである。つまり，CTCをうまく使うともっと時間を有効に使え，BASICのキー反応が向上するのである。それらに関しては自由研究としておきたい。

**リスト12-13　X1 turbo用テレビインリスト変更点(turbo BASIC用)**

```
180 TEMPO 80
190 MEM$(&HEF50,2)=HEXCHR$("ED 4D")
200 MEM$(&HF81E,2)=MKI$(&HEF13)
```

**リスト12-14　X1用テレビインリスト変更点(NEW BASICでは動かない)**

```
180 MEM$(&HEF50,2)=HEXCHR$("ED 4D")
190 POKE &H61AE,1 :'IF TAPE BASIC THEN &H68E2
200  CTC=&H704   :'CZ-8BS1
210 'CTC=&HA04   :'CZ-8BR1
220 'CTC=&H1FA8  :'CZ-8BM2
230 OUT CTC+0,3:'ﾁｬﾝﾈﾙ0 RESET
240 OUT CTC+1,3:'ﾁｬﾝﾈﾙ1 RESET
250 OUT CTC+2,3:'ﾁｬﾝﾈﾙ2 RESET
260 OUT CTC+3,3:'ﾁｬﾝﾈﾙ3 RESET
270 '
280 MEM$(&H5E,2)    =HEXCHR$("13 EF"):'ﾜﾘｺﾐ TABLE SET
290 '
300 OUT CTC+0,&B100111   :OUT CTC+0,30     :'ﾁｬﾝﾈﾙ0 SET
310 OUT CTC+0,&H58                         :'ﾜﾘｺﾐﾍﾞｸﾄﾙ SET
320 OUT CTC+3,&B11000111:OUT CTC+3,125    :'ﾁｬﾝﾈﾙ3 SET
```

# 第13章

# データレコーダ




テープもやってしまうのである

# 第13章
# テープもやってしまうのである・・

この章ではデータレコーダについてやるのであるが，最初に断っておくと，私はテープには未来がないと思っているのである。確かにテープは安価であるし，X1，MZ では信頼性があってボーレートも**まあまあである**。しかし，いかんせんディスクにはかなうべくもない。たとえばアセンブラを使おうなどと考えたなら，1バイトの修正に平気で 10 分以上かかってしまうのである（もちろんそうでないアセンブラもあるが，その場合はアセンブルできるオブジェクトの大きさが制限されたりするのだ)。そのように考えてみるなら，私が「う～む，これはテープのテクニックを極めたとしても，応用は限られてくるなぁ」と考えたとしても，もっともな話であろう。

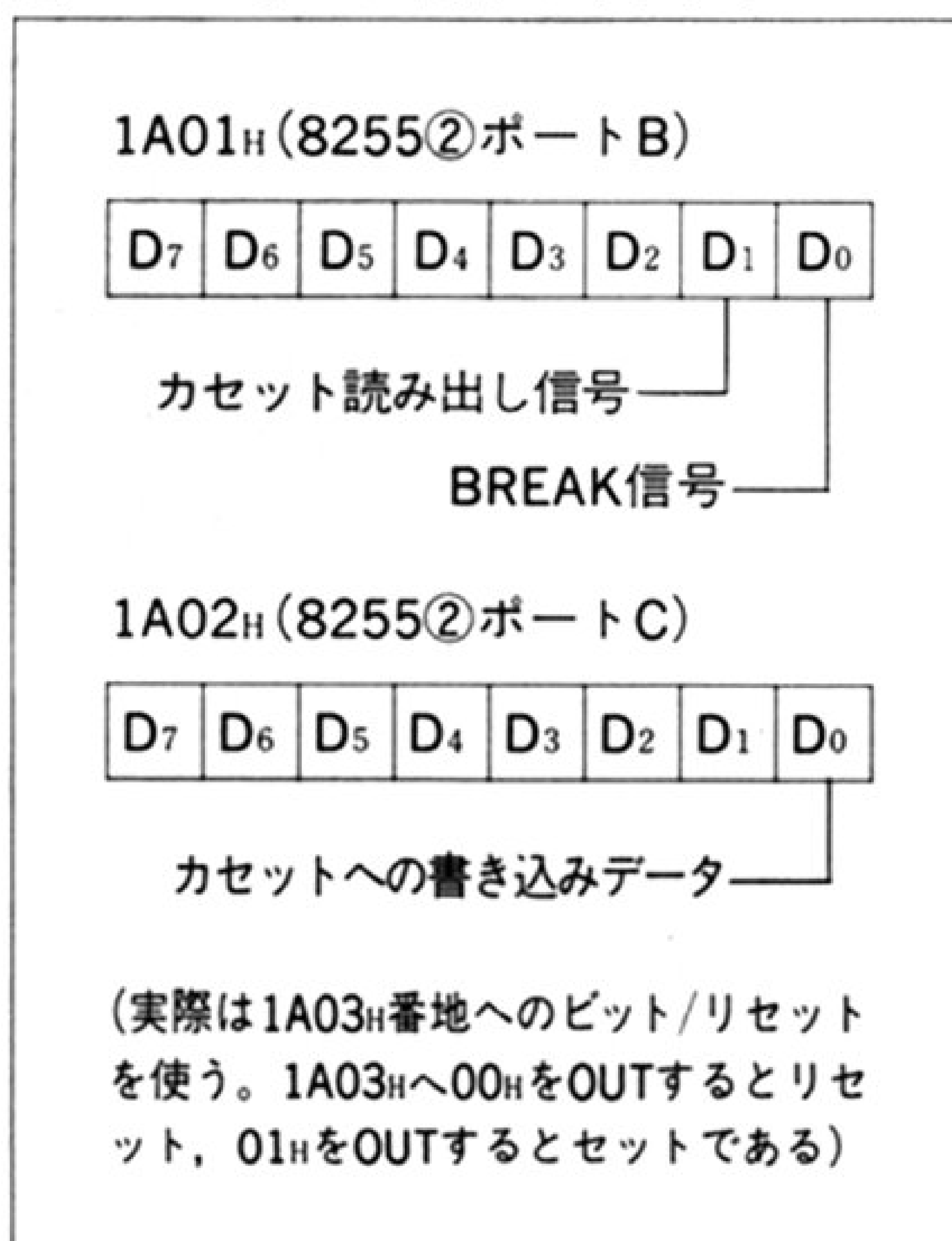
しかしこの世にはまだまだテープユーザーがいることでもあるから，やはりやってしまうのである。

## テープ関係のI/Oアドレスである

早い話が**図 13-1** である。

ここで注意が必要なのは，$1A01_H$番地の BREAK 信号である。ここはカセットが動作中に SHIFT ＋ BREAK（もしくは CTRL ＋ C）が押されたり，カセットコントロールキーが押されたりして動作がその筋したときなどに Low（＝0）になるのである。よってここをチェックしないと，カセットメカが停止しているにもかかわらず，プログラムの方はぽかんと口を開けてデータを待っているというだらしないことになってしまうのである。そのようなプログラムはフルロジックコントロール可能な X1 では **御法度** なのであ

図 13-1　テープ関係の I/O アドレス



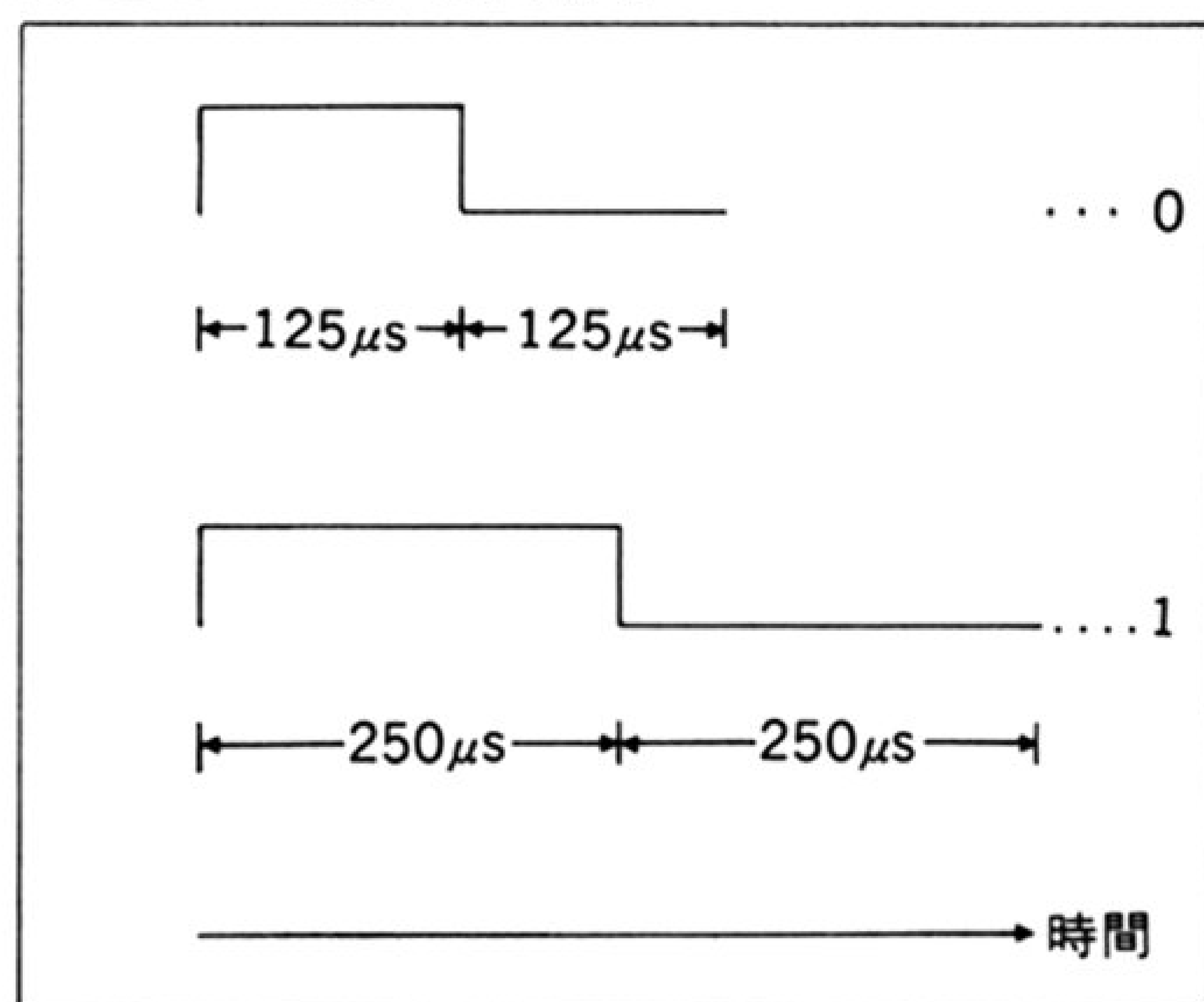
る。心得ていただきたい。

あと言っておくべきこととなると，サブCPUのことがあるが，詳しくは第4章を見ていただきたい。

# 0と1である

**図13-2**を見ていただきたい。これらがX1のテープにおける0と1である。0と1が記録できるなら，デジタルデータに限れば充分に戦闘可能になる。実際にはこれらのほかに0でも1でもないものもあるわけだが，それらは単なる「無記録部分」として扱われる。

図13-2 「0」と「1」の信号




では具体的に0，1を書く方法と読む方法を示すことにする。

まずは書く方である。図13-2に示すような信号をテープに記録するわけだから，適当にタイミングを取って「カセットテープへの書き込みデータ」をL→H→Lにしてやればよいのである。

次に読む方である。方法はいくらでも考えられるが，主流は，

**1)** 信号の立ち上がり（L→H）をとらえる

**2)** 185 μs 待つ

**3)** 信号を読み取ってLなら0，Hなら1

という手法である。読者の中には「立ち上がりから立ち下がりまでの時間を計ってもいいじゃないか」と考える人もいるだろうが，それはいけないのである。なぜかというと，それだとノイズを拾う確率が**どどーんと**増えてしまうからである。「1回のサンプリング当たりの平均エラーレートがこれこれだとすると，0と1を読み間違う確率があーたらこーたら」というよーなことは，若干なのでやめておくが，先程述べた主流の方法で世の中が丸く収まっているのだから，とりあえずはこれでいーのだ。しかし，広い世の中には「うんにゃ！　私はテープリードエラーで痛い目を見たことがある。読むアルゴリズムを変えれば，あのときのプログラムを回収できるかもしれない！」などと言う人もいるかもしれない。確かにそのとーりであろう。この本は，そのよーな人が**自力で**問題を解決するという基本姿勢を保ちつつ，さっさと次に行ってしまうのであった。

# テープの記録フォーマットである

「０」と「１」をテープに記録することによって，プログラムやデータを保存するわけであるが，やはり世の常としていきなりファイル名やデータを書くわけにはいかないのである。なぜならば，**テープはどこから読み始められるか分からないので，**「今読んでいるのはファイル名なのか？　それともデータの途中なのか？」ということを判別できなければならないからである。つまり「はいっ！　もうじき〝ファイル名〟の始まりですよっ！」とかいう**特別なマーク**が必要なのだ。そんなこんなの形式は各機種，各システムで違ってくるわけであるが，**図 13-3** が一番大事な X1 HuBASIC でのフォーマットである。MZ でのフォーマットも**図 13-4** に示しておいた。これらは「Oh！MZ」'84 年 7 月号に掲載されていたものである。

図 13-3　X1 HuBASIC でのテープフォーマット

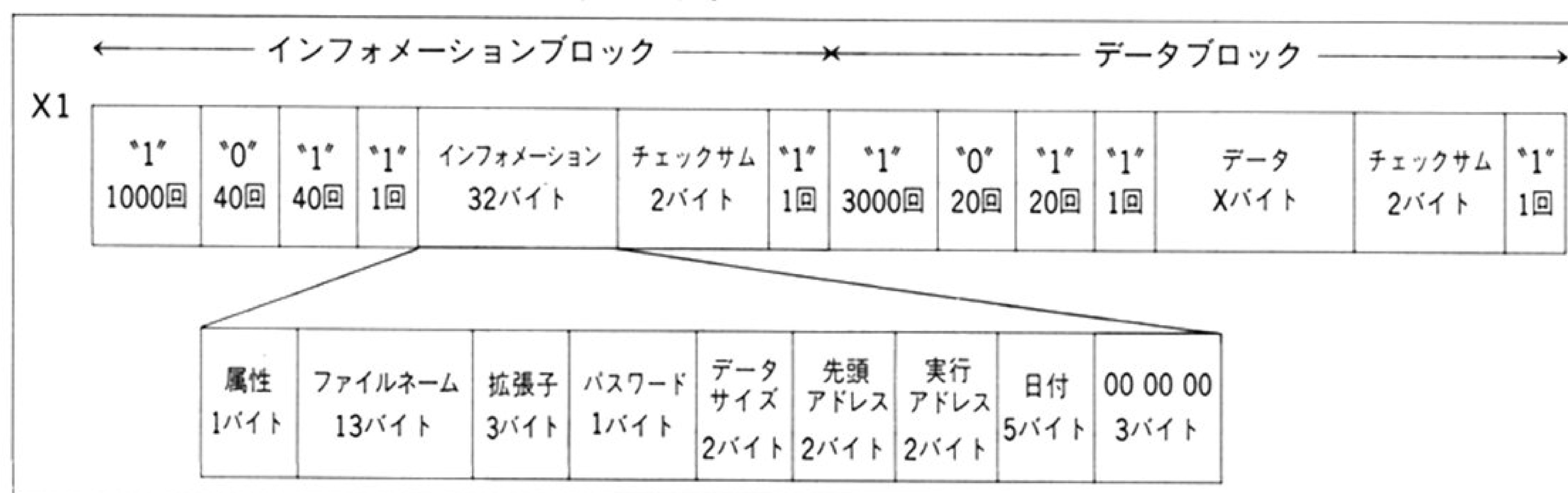
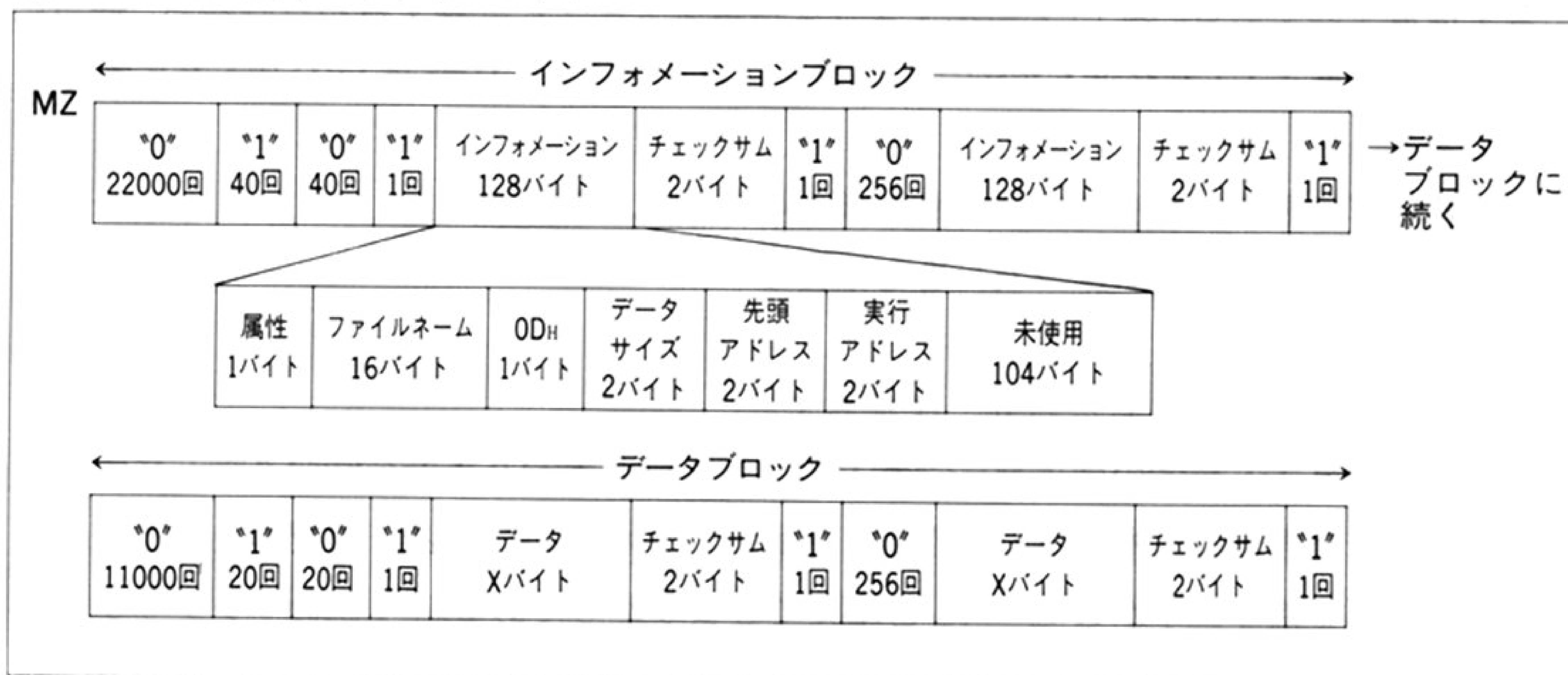



図 13-4　MZ でのテープフォーマット




図には書いてないが，実はまず最初に無録音部が 8 秒間ある。これはテープのリーダーの読み飛ばしと，APSS のために必要である。APSS，別名「一発頭出し」は簡単に言ってしまえば「早送りしながら無録音部を検出して止まる」ということだからである。続いてインフォメーション部とデータ部が来るわけである。これらの基本的な構造は同じであるからインフォメーション部だけをねっとりと説明する。

インフォメーション部は図 13-3 に示してあるような**マーク**から始まる。つまり「１」が

1000個，「0」が40個，また「1」が40個，最後に駄目押しで「1」が1個である。その後に続くのがインフォメーションの本体であるが，そのバイト数は32バイトと決まっている。

さて，そこで「1バイト」の書き方である。まず1バイトに先立って**スタートビット**なるものが書かれる。これは「1」が1回である。なぜこのよーなものがあるかというと，やはりこれはテープが宿命的に持つ悲しさと言わざるを得ない。つまり，テープはメカの回転変動などの影響を受けやすいので，所々に印を入れておかなければならないのだ。ただしこれを入れるとエラー発生率が下がるというものではなく，エラーの発生を検知できるというぐらいの役割しかない。それともう一つ，先程述べた**マーク**と同じパターンがテープに書かれるということを避ける意味もあるだろう。9ビットごとに「1」が来るのだから，ファイル名を書いている最中に偶然マークと同じパターン（「0」が続けて20個）が現れてしまう心配はないのである(こちらの方がもっと大事であろう)。そのよーな手続きを経た後に，1バイトを書くのである。書く順番は第7ビットからである。

このようにして32バイトのインフォメーションを書いた後に続いて，2バイトのチェックサムがある。これは「チェックサムエラー」を出すためである。図13-3にもあるように，これはインフォメーションブロックにある32バイトのデータ中にある「1」の個数である。この値は変態なことに，High，Lowの順になっている。つまり「1」が300個＝$12C_H$個だったなら，この2バイトは$01_H$，$2C_H$の順に並んでいる(インフォメーションの場合は32バイトだから，最大でも$100_H$であるが)。Z80での通常の2バイト数値とは逆順なのである。

インフォメーション部は最後に「1」が1回書かれて終わる。これはストップビットと呼ばれる。

データブロックは，マークが少し違うこと，長さ（バイト数）が不定であることを除けばインフォメーションブロックと同じである。データブロックのバイト数はインフォメーションブロックの18，19バイト目（「0バイト目」から数える）にあるから，このことを心得ていれば問題はない。

# 読んだり書いたりするのである

ここからプログラムの解説モードに突入する。

**リスト13-1**はテープのリード/ライトを行なう機械語プログラム，**リスト13-2**はそのダンプリスト，**リスト13-3**がそれを使っているプログラムである。ただし恐ろしいことに，普通では満足できないというその筋の「さが」によって，さまざまな仕掛けがしてあるのだ。すなわち，

- ボーレートを自由に設定できる
- G-RAMを使って64Kバイトを読み書きできる
- フォーマットはMZかX1を選択できる

となっているのである。その結果として使い方は少々その筋しているが，パターンは限ら

れているのだからこれでいーのだ。

**リスト 13-1 テープ入出力プログラム**

```
                                .Z80
                                .PHASE  0BC00H
                        ;
1A01                    FMCMT   EQU     1A01H
1A03                    C8255   EQU     1A03H
1A02                    TOCMT   EQU     1A02H
                        ;
0072                    PATRMM  EQU     0072H   ;LD     (HL),D:NOP
51ED                    PATRIO  EQU     51EDH   ;OUT    (C),D
                        ;
0056                    PATWMM  EQU     0056H   ;LD     D,(HL):NOP
50ED                    PATWIO  EQU     50EDH   ;IN     D,(C)
                        ;
0041                    X1L     EQU     65
001D                    X1S     EQU     29
002E                    X1W     EQU     46
                        ;
                        ;4 JUMP TABLES
                        ;
BC00    C3 BC27         WHEAD:  JP      WHEAD1
BC03    C3 BC4A         WBODY:  JP      WBODY1
BC06    C3 BD30         RHEAD:  JP      RHEAD1
BC09    C3 BD47         RBODY:  JP      RBODY1
                        ;
BC0C    BE60            FCBAD:  DW      LAST
BC0E    0000            BDYLEN: DW      0000H
                        ;
BC10    00              FLAG:   DB      0       ;MZ or X1
BC11    41              LONGH:  DB      X1L     ;LONG HIGH
BC12    33              LONGL:  DB      X1L-14  ;LONG LOW
BC13    1D              SHORTH: DB      X1S     ;SHORT HIGH
BC14    0F              SHORTL: DB      X1S-14  ;SHORT L
BC15    2E              W185:   DB      X1W     ;SAMPLING TIME
BC16    05              AFTERW: DB      5       ;SKIP
                        ;
BC17                    ISET:                   ;FCB MARK
BC17    03E8            IMARK1: DW      1000
BC19    0028            IMARK2: DW      40
BC1B    0028            IMARK3: DW      40
                        ;
BC1D                    DSET:                   ;BODY MARK
BC1D    0BB8            DMARK1: DW      3000
BC1F    0014            DMARK2: DW      20
BC21    0014            DMARK3: DW      20
                        ;
BC23    0020            FCBLEN: DW      32      ;FCB LENGTH
                        ;
BC25                    SUM:    DS      2       ;WORK AREA
                        ;
                        ;***** WRITE ROUTINE BEGIN *****
                        ;
BC27    AF              WHEAD1: XOR     A       ;WRITE HEAD
BC28    01 1A03                 LD      BC,C8255;8255 CONTROLE
BC2B    ED 79                   OUT     (C),A   ;WRITE 'L'
BC2D    21 21E4                 LD      HL,8676 ;COUNTER
BC30    CD BE3F                 CALL    WAITXX  ;WRITE 8 SEC GAP
                        ;
BC33    F3                      DI
BC34    21 0056                 LD      HL,PATWMM
BC37    22 BCBB                 LD      (PATW),HL       ;PATCH!
BC3A    2A BC0C                 LD      HL,(FCBAD)
BC3D    ED 5B BC23              LD      DE,(FCBLEN)
BC41    FD 21 BC17              LD      IY,ISET
BC45    CD BC6A                 CALL    WRITGO  ;WRITE HEAD
BC48    FB                      EI
BC49    C9                      RET
                        ;
BC4A    21 0100         WBODY1: LD      HL,100H ;WRITE BODY
BC4D    CD BE3F                 CALL    WAITXX  ;WRITE GAP
                        ;
BC50    21 50ED                 LD      HL,PATWIO
BC53    22 BCBB                 LD      (PATW),HL       ;PATCH!
BC56    21 FF00                 LD      HL,0000H-100H   ;0FF00H
BC59    01 4000                 LD      BC,4000H        ;GRAM
BC5C    ED 5B BC0E              LD      DE,(BDYLEN)
BC60    FD 21 BC1D              LD      IY,DSET
```

```
BC64    F3                    DI
BC65    CD BC6A               CALL    WRITGO  ;WRITE BODY
BC68    FB                    EI
BC69    C9                    RET
                      ;
BC6A    D9            WRITGO: EXX
BC6B    FD 4E 00              LD      C,(IY+0)
BC6E    FD 46 01              LD      B,(IY+1)
BC71    37                    SCF             ;CARRY=1
BC72    CD BCE3               CALL    WMARK   ;WRITE MARK1
                      ;
BC75    FD 4E 02              LD      C,(IY+2)
BC78    FD 46 03              LD      B,(IY+3)
BC7B    B7                    OR      A       ;CARRY=0
BC7C    CD BCE3               CALL    WMARK   ;WRITE MARK2
                      ;
BC7F    FD 4E 04              LD      C,(IY+4)
BC82    FD 46 05              LD      B,(IY+5)
BC85    37                    SCF             ;CARRY=1
BC86    CD BCE3               CALL    WMARK   ;WRITE MARK3
                      ;
BC89    37                    SCF             ;CARRY=1
BC8A    CD BCFB               CALL    WBIT    ;START BIT
                      ;
BC8D    D9                    EXX
BC8E    CD BC96               CALL    WBYTES  ;HONTAI
                      ;
BC91    37                    SCF             ;CARRY=1
BC92    CD BCFB               CALL    WBIT    ;STOP BIT
BC95    C9                    RET
                      ;
BC96    D9            WBYTES: EXX
BC97    21 0000               LD      HL,0000H ;CHECK SUM
BC9A    D9                    EXX
BC9B    CD BCB5               CALL    WBS1
                      ;
BC9E    D9                    EXX
BC9F    7C                    LD      A,H
BCA0    65                    LD      H,L
BCA1    6F                    LD      L,A     ;SWAP H-L
BCA2    22 BC25               LD      (SUM),HL
BCA5    11 0002               LD      DE,2
BCA8    21 0056               LD      HL,PATWMM
BCAB    22 BCBB               LD      (PATW),HL       ;PATCH!
BCAE    21 BC25               LD      HL,SUM
BCB1    CD BCB5               CALL    WBS1    ;WRITE CHECK SUM
BCB4    C9                    RET
                      ;
BCB5    D5            WBS1:   PUSH    DE
BCB6    78                    LD      A,B
BCB7    FE 40                 CP      40H     ;BC < 4000H ?
BCB9    38 05                 JR      C,MRAMW ;THEN JUMP
BCBB    ED 50         PATW:   IN      D,(C)   ;GET FROM GRAM
BCBD    C3 BCC1               JP      MRAMW1
BCC0    56            MRAMW:  LD      D,(HL)  ;GET FROM MRAM
BCC1    23            MRAMW1: INC     HL
BCC2    03                    INC     BC      ;INC POINTERS
BCC3    CD BCCD               CALL    WBYTE   ;WRITE 1 BYTE
BCC6    D1                    POP     DE
BCC7    1B                    DEC     DE      ;DEC COUNTER
BCC8    7A                    LD      A,D
BCC9    B3                    OR      E       ;CHECK IT
BCCA    20 E9                 JR      NZ,WBS1
BCCC    C9                    RET
                      ;
BCCD    C5            WBYTE:  PUSH    BC      ;WRITE 1 BYTE
BCCE    37                    SCF
BCCF    CD BCFB               CALL    WBIT
                      ;
BCD2    06 08                 LD      B,8     ;8 BITS
BCD4    CB 12         WBYTE1: RL      D
BCD6    D9                    EXX
BCD7    D2 BCDB               JP      NC,WBYTE2
BCDA    23                    INC     HL      ;INC CHECK SUM
BCDB    D9            WBYTE2: EXX
BCDC    CD BCFB               CALL    WBIT    ;WRITE 1 BIT
BCDF    10 F3                 DJNZ    WBYTE1
BCE1    C1                    POP     BC
BCE2    C9                    RET
                      ;
BCE3    F5            WMARK:  PUSH    AF      ;SAVE CARRY
```

```
BCE4    3A BC10                    LD      A,(FLAG);GET FLAG
BCE7    CE 00                      ADC     A,00H
BCE9    1F                         RRA     ;IF MZ CY REVERSE
BCEA    F5               WMARK1:   PUSH    AF
BCEB    C5                         PUSH    BC
BCEC    CD BCFB                    CALL    WBIT
BCEF    C1                         POP     BC
BCF0    0B                         DEC     BC
BCF1    78                         LD      A,B
BCF2    B1                         OR      C        ;COUNTER
BCF3    28 03                      JR      Z,WMARK2
BCF5    F1                         POP     AF
BCF6    18 F2                      JR      WMARK1
                         ;
BCF8    F1               WMARK2:   POP     AF
BCF9    F1                         POP     AF
BCFA    C9                         RET
                         ;
BCFB                     WBIT:     ;CY->"1",NON CY->"0"
BCFB    C5                         PUSH    BC               ;11
BCFC    30 09                      JR      NC,WBIT0         ;12 or 7
BCFE    3A BC12                    LD      A,(LONGL)        ;13
BD01    08                         EX      AF,AF'           ;4
BD02    3A BC11                    LD      A,(LONGH)        ;13
BD05    18 0B                      JR      WBIT1            ;12
BD07    38 09            WBIT0:    JR      C,WBIT1          ;12 or 7
BD09    3A BC14                    LD      A,(SHORTL)       ;13
BD0C    08                         EX      AF,AF'           ;4
BD0D    3A BC13                    LD      A,(SHORTH)       ;13
BD10    18 00                      JR      WBIT1            ;12
BD12    01 1A03          WBIT1:    LD      BC,C8255         ;10
BD15    F5                         PUSH    AF               ;11
BD16    3E 01                      LD      A,1              ;7
                         ;
BD18    ED 79                      OUT     (C),A    ;12
BD1A    F1                         POP     AF       ;10
BD1B    CD BE52                    CALL    WAITX    ;17
BD1E    AF                         XOR     A        ;4
                         ;
BD1F    ED 79                      OUT     (C),A    ;12
BD21    08                         EX      AF,AF'   ;4
BD22    CD BE52                    CALL    WAITX    ;17
BD25    01 1A01                    LD      BC,FMCMT;10
BD28    ED 78                      IN      A,(C)    ;12
BD2A    0F                         RRCA             ;4
BD2B    D2 BE0D                    JP      NC,BREAK;10
BD2E    C1                         POP     BC       ;10
BD2F    C9                         RET              ;10
                         ;
                         ;***** WRITE ROUTINE END *****
                         ;
                         ;***** READ ROUTINE BEGIN *****
                         ;
BD30    F3               RHEAD1:   DI               ;READ HEAD
BD31    21 0072                    LD      HL,PATRMM
BD34    22 BDC6                    LD      (PATR),HL        ;PATCH!
BD37    2A BC0C                    LD      HL,(FCBAD)       ;DCB ADD.
BD3A    ED 5B BC23                 LD      DE,(FCBLEN)      ;LENGTH
BD3E    FD 21 BC17                 LD      IY,ISET
BD42    CD BD61                    CALL    READGO
BD45    FB                         EI
BD46    C9                         RET
                         ;
BD47    F3               RBODY1:   DI                       ;READ BODY
BD48    21 51ED                    LD      HL,PATRIO
BD4B    22 BDC6                    LD      (PATR),HL        ;PATCH!
BD4E    21 FF00                    LD      HL,0000H-100H    ;DATA ADD.
BD51    01 4000                    LD      BC,4000H
BD54    ED 5B BC0E                 LD      DE,(BDYLEN)      ;LENGTH
BD58    FD 21 BC1D                 LD      IY,DSET
BD5C    CD BD61                    CALL    READGO
BD5F    FB                         EI
BD60    C9                         RET
                         ;
BD61    D9               READGO:   EXX
BD62    CD BDFD          REDGO1:   CALL    COUNT0   ;COUNT "0"
BD65    FD 5E 02                   LD      E,(IY+2)
BD68    FD 56 03                   LD      D,(IY+3)
BD6B    B7                         OR      A
BD6C    ED 52                      SBC     HL,DE    ;CHECK MARK2
BD6E    20 F2                      JR      NZ,REDGO1
```

```
BD70    FD 46 04                  ;
                                          LD      B,(IY+4)
BD73    05                                DEC     B
BD74    3A BC10                           LD      A,(FLAG)
BD77    57                                LD      D,A
BD78    CD BE14                   REDGO2: CALL    RBIT
BD7B    3E 00                             LD      A,00H
BD7D    17                                RLA             ;X1->1,MZ->0
BD7E    BA                                CP      D       ;BIT = FLAG ?
BD7F    CA BE0D                           JP      Z,BREAK ;THEN BREAK
BD82    10 F4                             DJNZ    REDGO2
                                  ;
BD84    CD BE14                           CALL    RBIT    ;READ LAST MARK
BD87    D2 BD62                           JP      NC,REDGO1
                                  ;
BD8A    D9                                EXX
BD8B    CD BDA2                           CALL    RBYTES  ;READ HONTAI
                                  ;
BD8E    ED 5B BC25                        LD      DE,(SUM)
BD92    7A                                LD      A,D
BD93    53                                LD      D,E
BD94    5F                                LD      E,A     ;SWAP H-L
BD95    B7                                OR      A
BD96    ED 52                             SBC     HL,DE   ;CHECK SUM
BD98    C2 BE11                           JP      NZ,CSERR
                                  ;
BD9B    CD BE14                           CALL    RBIT    ;STOP BIT
BD9E    D2 BE0D                           JP      NC,BREAK
                                  ;
BDA1    C9                                RET
                                  ;
BDA2    D9                        RBYTES: EXX
BDA3    21 0000                           LD      HL,0000H ;CHECK SUM
BDA6    D9                                EXX
BDA7    CD BDBD                           CALL    RBS1    ;READ HONTAI
                                  ;
BDAA    D9                                EXX
BDAB    E5                                PUSH    HL
BDAC    11 0002                           LD      DE,2
BDAF    21 0072                           LD      HL,PATRMM
BDB2    22 BDC6                           LD      (PATR),HL       ;PATCH!
BDB5    21 BC25                           LD      HL,SUM
BDB8    CD BDBD                           CALL    RBS1    ;READ CHECK SUM
BDBB    E1                                POP     HL
BDBC    C9                                RET
                                  ;
BDBD    D5                        RBS1:   PUSH    DE
BDBE    CD BDD5                           CALL    RBYTE
BDC1    78                                LD      A,B
BDC2    FE 40                             CP      40H     ;BC < 4000H ?
BDC4    38 05                             JR      C,MRAMR ;THEN JUMP
BDC6    ED 51                     PATR:   OUT     (C),D   ;STORE TO GRAM
BDC8    C3 BDCC                           JP      MRAMR1
BDCB    72                        MRAMR:  LD      (HL),D  ;STORE TO MRAM
BDCC    23                        MRAMR1: INC     HL
BDCD    03                                INC     BC
BDCE    D1                                POP     DE
BDCF    1B                                DEC     DE
BDD0    7A                                LD      A,D
BDD1    B3                                OR      E       ;CHECK COUNTER
BDD2    20 E9                             JR      NZ,RBS1
BDD4    C9                                RET
                                  ;
BDD5    C5                        RBYTE:  PUSH    BC      ;READ 1 BYTE
BDD6    CD BE14                           CALL    RBIT
BDD9    30 32                             JR      NC,BREAK        ;START BIT
                                  ;
BDDB    16 00                             LD      D,00H   ;CLEAR DATA
BDDD    06 08                             LD      B,8
BDDF    CD BE14                   RBYTE1: CALL    RBIT
BDE2    D9                                EXX
BDE3    30 01                             JR      NC,RBYTE2
BDE5    23                                INC     HL      ;INC CHECK SUM
BDE6    D9                        RBYTE2: EXX
BDE7    CB 12                             RL      D       ;STORE 1 BIT
BDE9    10 F4                             DJNZ    RBYTE1
BDEB    C1                                POP     BC
BDEC    C9                                RET
                                  ;
BDED    21 0000                   COUNT1: LD      HL,0000H
BDF0    3A BC10                           LD      A,(FLAG)
```

```
BDF3    B7                       OR      A
BDF4    20 10                    JR      NZ,CT01 ;IF MZ COUNT0
BDF6    CD BE14         CT11:    CALL    RBIT
BDF9    D0                       RET     NC
BDFA    23                       INC     HL
BDFB    18 F9                    JR      CT11
                        ;
BDFD    21 0000         COUNT0:  LD      HL,0000H
BE00    3A BC10                  LD      A,(FLAG)
BE03    B7                       OR      A
BE04    20 F0                    JR      NZ,CT11 ;IF MZ COUNT1
BE06    CD BE14         CT01:    CALL    RBIT
BE09    D8                       RET     C
BE0A    23                       INC     HL
BE0B    18 F9                    JR      CT01
                        ;
BE0D    3E 1D           BREAK:   LD      A,29     ;ERR CODE
BE0F    DD E9                    JP      (IX)
BE11    AF              CSERR:   XOR     A        ;UNPRINTABLE
BE12    DD E9                    JP      (IX)
                        ;
BE14    C5              RBIT:    PUSH    BC
BE15    01 1A01                  LD      BC,FMCMT
BE18    ED 78           RBIT1:   IN      A,(C)    ;IN DATA
BE1A    0F                       RRCA
BE1B    30 F0                    JR      NC,BREAK
BE1D    0F                       RRCA
BE1E    38 F8                    JR      C,RBIT1 ;WHILE(HIGH) DO
                        ;
BE20    ED 78           RBIT2:   IN      A,(C)            ;12
BE22    0F                       RRCA                     ;4
BE23    D2 BE0D                  JP      NC,BREAK         ;10
BE26    0F                       RRCA                     ;4
BE27    D2 BE20                  JP      NC,RBIT2         ;10
                                                ;WHILE(LOW) DO
                        ;
                        ;NOW I GOT 'EDGE' !
BE2A    3A BC15                  LD      A,(W185)         ;7
BE2D    CD BE52                  CALL    WAITX            ;17
BE30    ED 78                    IN      A,(C)
BE32    0F                       RRCA
BE33    D2 BE0D                  JP      NC,BREAK
BE36    0F                       RRCA
BE37    3A BC16                  LD      A,(AFTERW)       ;SKIP TIME
BE3A    DC BE52                  CALL    C,WAITX
BE3D    C1                       POP     BC
BE3E    C9                       RET
                        ;
                        ;***** READ ROUTINE END *****
                        ;
                        ;***** WAIT ROUTINE *****
                        ;
BE3F    01 1A01         WAITXX:  LD      BC,FMCMT;10
BE42    ED 78           WAPSS:   IN      A,(C)    ;12
BE44    0F                       RRCA             ;4
BE45    D2 BE0D                  JP      NC,BREAK;10
BE48    AF                       XOR     A        ;4
BE49    CD BE52                  CALL    WAITX    ;17+14*256+31
BE4C    2B                       DEC     HL       ;6
BE4D    7C                       LD      A,H      ;4
BE4E    B5                       OR      L        ;4
BE4F    20 F1                    JR      NZ,WAPSS ;12 or 7
BE51    C9                       RET              ;10
                        ;DAITAI HL*3688
BE52    F5              WAITX:   PUSH    AF               ;11
BE53    3D              WAITXL:  DEC     A                ;4
BE54    C2 BE53                  JP      NZ,WAITXL        ;10
                        ;
BE57    F1                       POP     AF               ;10
BE58    C9                       RET                      ;10
                        ;11+14*Acc+20
                        ;Acc=46->644+31=675
                        ;
        MACRO-80 3.44   09-Dec-81        PAGE    1-7


BE59                             DS      7
BE60                    LAST:    DS      0
                                 END
```

### リスト 13-2 テープ入出力ダンプリスト

```
BC00 C3 27 BC C3 4A BC C3 30 : 62
BC08 BD C3 47 BD 60 BE 00 00 : A2
BC10 00 41 33 1D 0F 2E 05 E8 : BB
BC18 03 28 00 28 00 B8 0B 14 : 2A
BC20 00 14 00 20 00 1A 1A AF : 17
BC28 01 03 1A ED 79 21 E4 21 : AA
BC30 CD 3F BE F3 21 56 00 22 : 56
BC38 BB BC 2A 0C BC ED 5B 23 : D4
BC40 BC FD 21 17 BC CD 6A BC : A0
BC48 FB C9 21 00 01 CD 3F BE : B0
BC50 21 ED 50 22 BB BC 21 00 : 18
BC58 FF 01 00 40 ED 5B 0E BC : 52
BC60 FD 21 1D BC F3 CD 6A BC : DD
BC68 FB C9 D9 FD 4E 00 FD 46 : 2B
BC70 01 37 CD E3 BC FD 4E 02 : F1
BC78 FD 46 03 B7 CD E3 BC FD : 66
---------------------------------
SUM: D9 80 90 9D 3E 3C 75 78 C262

BC80 4E 04 FD 46 05 37 CD E3 : 81
BC88 BC 37 CD FB BC D9 CD 96 : B3
BC90 BC 37 CD FB BC C9 D9 21 : 3A
BC98 00 00 D9 CD B5 BC D9 7C : 6C
BCA0 65 6F 22 25 BC 11 02 00 : EA
BCA8 21 56 00 22 BB BC 21 25 : 56
BCB0 BC CD B5 BC C9 D5 78 FE : 0E
BCB8 40 38 05 ED 50 C3 C1 BC : FA
BCC0 56 23 03 CD CD BC D1 1B : BE
BCC8 7A B3 20 E9 C9 C5 37 CD : C8
BCD0 FB BC 06 08 CB 12 D9 D2 : 4D
BCD8 DB BC 23 D9 CD FB BC 10 : 27
BCE0 F3 C1 C9 F5 3A 10 BC CE : 46
BCE8 00 1F F5 C5 CD FB BC C1 : 1E
BCF0 0B 78 B1 28 03 F1 18 F2 : 5A
BCF8 F1 F1 C9 C5 30 09 3A 12 : F5
---------------------------------
SUM: DD D3 D0 37 2A 8D 0F 52 7B60

BD00 BC 08 3A 11 BC 18 0B 38 : 26
BD08 09 3A 14 BC 08 3A 13 BC : 24
BD10 18 00 01 03 1A F5 3E 01 : 6A
BD18 ED 79 F1 CD 52 BE AF ED : D0
BD20 79 08 CD 52 BE 01 01 1A : 7A
BD28 ED 78 0F D2 0D BE C1 C9 : 9B
BD30 F3 21 72 00 22 C6 BD 2A : 55
BD38 0C BC ED 5B 23 BC FD 21 : 0D
BD40 17 BC CD 61 BD FB C9 F3 : 75
BD48 21 ED 51 22 C6 BD 21 00 : 25
BD50 FF 01 00 40 ED 5B 0E BC : 52
BD58 FD 21 1D BC CD 61 BD FB : DD
BD60 C9 D9 CD FD BD FD 5E 02 : 86
BD68 FD 56 03 B7 ED 52 20 F2 : 5E
BD70 FD 46 04 05 3A 10 BC 57 : A9
BD78 CD 14 BE 3E 00 17 BA CA : 78
---------------------------------
SUM: F3 6C 48 92 61 30 30 CF D3FE

BD80 0D BE 10 F4 CD 14 BE D2 : 40
BD88 62 BD D9 CD A2 BD ED 5B : 6C
BD90 25 BC 7A 53 5F B7 ED 52 : 03
BD98 C2 11 BE CD 14 BE D2 0D : 0F
BDA0 BE C9 D9 21 00 00 D9 CD : 27
BDA8 BD BD D9 E5 11 02 00 21 : 6C
BDB0 72 00 22 C6 BD 21 25 BC : 19
BDB8 CD BD BD E1 C9 D5 CD D5 : 68
BDC0 BD 78 FE 40 38 05 ED 51 : EE
BDC8 C3 CC BD 72 23 03 D1 1B : D0
BDD0 7A B3 20 E9 C9 C5 CD 14 : A5
BDD8 BE 30 32 16 00 06 08 CD : 11
BDE0 14 BE D9 30 01 23 D9 CB : A3
BDE8 12 10 F4 C1 C9 21 00 00 : C1
BDF0 3A 10 BC B7 20 10 CD 14 : CE
BDF8 BE D0 23 18 F9 21 00 00 : E3
---------------------------------
SUM: E6 60 6B FF 80 86 6E 37 795A

BE00 3A 10 BC B7 20 F0 CD 14 : AE
BE08 BE D8 23 18 F9 3E 1D DD : 02
BE10 E9 AF DD E9 C5 01 01 1A : 3F
BE18 ED 78 0F 30 F0 0F 38 F8 : D3
BE20 ED 78 0F D2 0D BE 0F D2 : F2
BE28 20 BE 3A 15 BC CD 52 BE : C6
BE30 ED 78 0F D2 0D BE 0F 3A : 5A
BE38 16 BC DC 52 BE C1 C9 01 : 49
BE40 01 1A ED 78 0F D2 0D BE : 2C
BE48 AF CD 52 BE 2B 7C B5 20 : 08
BE50 F1 C9 F5 3D C2 53 BE F1 : B0
BE58 C9 00 00 00 00 00 00 00 : C9
---------------------------------
SUM: 48 29 33 66 5E E9 DC 9D A9B4
```

### リスト 13-3 BASIC から「リスト 13-2」を使ってみる

```
100 CLEAR &HBC00
110 DEFUSR0=&HBC00:'WRITE FCB
120 DEFUSR1=&HBC03:'WRITE BODY
130 DEFUSR2=&HBC06:'READ FCB
140 DEFUSR3=&HBC09:'READ BODY
150 '
160 INIT
170 IF MEM$(&HBC00,3)<>HEXCHR$("C3 27 BC")THEN LOADM "TAPE.OBJ"
180 '
190 'for X1:ﾊｲｲﾛ ﾉ 2700 ﾎﾞｰ
200  MEM$(&HBC10,16)=HEXCHR$("00 41 33 1D 0F 2E 05 E8 03 28 00 28 00 B8 0B 14")
210  MEM$(&HBC20,5) =HEXCHR$("00 14 00 20 00")
220 '
230 'for S-OS "T:":2400 ﾎﾞｰ ﾅﾉﾆ X1 ﾄ ｵﾅｼﾞ ﾎﾞｰﾚｰﾄ ﾀﾞｯﾀﾘ ｽﾙ
240 'MEM$(&HBC10,16)=HEXCHR$("01 41 33 1D 0F 2E 01 F0 55 28 00 28 00 F8 2A 14")
250 'MEM$(&HBC20,5) =HEXCHR$("00 14 00 80 00")
260 '
270 'for MZ-80B/2000,2200:2000 ﾎﾞｰ
280 'MEM$(&HBC10,16)=HEXCHR$("01 58 56 28 26 40 05 F0 55 28 00 28 00 F8 2A 14")
```

```
290 'MEM$(&HBC20,5) =HEXCHR$("00 14 00 80 00")
300 '
310 'for MZ-80K/C/1200/700/1500:1200 ﾎﾞｰ
320 'MEM$(&HBC10,16)=HEXCHR$("01 83 81 3E 3C 5A 05 F0 55 28 00 28 00 F8 2A 14")
330 'MEM$(&HBC20,5) =HEXCHR$("00 14 00 80 00")
340 '
350 INPUT "R/W";A$:IF A$="R" THEN GOSUB"READ":GOTO 350
360 IF A$="W" THEN GOSUB"WRITE":GOTO 350
370 GOTO 350
380 '
390 LABEL"READ"
400 CMT=2:'PLAY
410 D$=USR2(""):'READ FCB
420 CMT=1:'CSTOP
430 PRINT #0 "FOUND",MEM$(&HBE61,16),
440 L=CVI(MEM$(&HBE60+18,2)):'LENGTH
450 S=CVI(MEM$(&HBE60+20,2)):'START ADDRESS
460 X=CVI(MEM$(&HBE60+22,2)):'JUMP ADDRESS
470 PRINT #0 RIGHT$("000"+HEX$(S),4),
480 PRINT #0 RIGHT$("000"+HEX$(S+L-1),4),
490 PRINT #0 RIGHT$("000"+HEX$(X),4)
500 '
510 MEM$(&HBC0E,2)=MEM$(&HBE60+18,2):'COPY LENGTH
520 CMT=2:'PLAY
530 PRINT #0 "LOADING",MEM$(&HBE61,16)
540 D$=USR3(""):'READ BODY
550 CMT=1:'CSTOP
560 RETURN
570 '
580 LABEL"WRITE"
590 PRINT #0 "WRITING",MEM$(&HBE61,16)
600 CMT=10:'WRITE
610 D$=USR0(""):'WRITE FCB
620 CMT=1:'CSTOP
630 '
640 MEM$(&HBC0E,2)=MEM$(&HBE60+18,2):'COPY LENGTH
650 CMT=10:'PLAY
660 D$=USR1(""):'WRITE BODY
670 CMT=1:'CSTOP
680 RETURN
```

では機械語プログラムは後にして，先にリスト13-3の使い方を説明する。

よくある展開として，各モードの指定は「MEM$」で機械語ルーチンにパラメータを渡すことで行なっている。一番最初に指定してあるのはX1フォーマットの場合のパラメータである。大抵の用はこれで済むだろうが，それだけでは面白くないので，後に続くようにさまざまなボーレートとフォーマットの挑戦を受けているのである。使うときはREM(')を適当に入れたり消したりしていただきたい。

パラメータの領域は$BC10_H$〜$BC24_H$になっている。それぞれの意味は**表13-1**に示してある。この部分はリスト13-1と併せて見た方が分かりやすいだろう。

それはともかく，このリスト13-3は30Kバイト以上の長さのファイル(実はBASIC)を読み書きさせて試験してあるので，異機種のフォーマットも大丈夫のはずである。ただし1200bpsだけは例外で，CZ-8RL1を使った場合は，**背面にあるボーレート切り換えスイッチをLowにしておけばOK**だが，それ以外の内蔵デッキでは，かなりエラーが出てしまうようである。ちなみに私は，しばらくの間このボーレート切り換えスイッチに気が付かず，バグ地獄をはいずり回っていたのであった。

それはさておき，具体的な説明をしておく。110〜140行にDEFUSR文が4個並んでいる。機能は注釈にあるとおりである。まずは390行からのテープリードであるが，最初にCMT命令でカセットデッキをREAD状態にしている。プログラム作成中は機械語ルーチンの中にカセットデッキコントロールのためのサブルーチンも入れてあったのだが，ちょ

**表 13-1　リスト 1 のパラメータの説明**

BC10$_H$＝フラグである。0ならX1フォーマット，1ならMZフォーマット。

BC11$_H$＝“1”のHighの長さ。ウエイトルーチンに渡すパラメータである。
BC12$_H$＝“1”のLowの長さ。
BC13$_H$＝“0”のHighの長さ。
BC14$_H$＝“0”のLowの長さ。
BC15$_H$＝信号の立ち上がりからサンプリングまでの待ち時間（X1では185$\mu$s)。
BC16$_H$＝“1”を読んだ後のスキップ時間（あんまりたいした意味はない)。

BC17, 8$_H$＝インフォメーションブロック（FCB）の最初のマークの数。
BC19, A$_H$＝インフォメーションブロックの2番目のマークの数。
BC1B, C$_H$＝インフォメーションブロックの3番目のマークの数。

BC1D, E$_H$＝データブロックの最初のマークの数。
BC1F, 20$_H$＝データブロックの2番目のマークの数。
BC21, 2$_H$＝データブロックの3番目のマークの数。
BC23, 4$_H$＝インフォメーションブロックの長さ。

っと長いし，第 4 章でやったサブ CPU の使い方そのものなので，取り除いてしまったのである。次に来るのは「USR2 (“　”)」で，これはインフォメーションブロックの読み込みである。読み込む位置（アドレス）は BC0C$_H$ からの 2 バイトに入っていることになっている。何もいじらなければここには BE60$_H$が入っているので，モニタでダンプすれば見ることができるのである。読み込み終わったなら，一旦テープを止めて（CMT＝1）ファイル名を表示する。時々ファイル名の中に変なコントロールコードが交じっていることがあるので，「PRINT　#0」を使っている。ここではさらに縁起ものなのでスタートアドレス，エンドアドレス，実行アドレスも表示している。

次に「USR3 (“　”)」でファイルの本体を読み込むわけであるが，ここで先程言っておいた**長さ**が出てくる。長さは BC0E$_H$ からの 2 バイトで指定するのだ。サンプルプログラムでは常識どおりにインフォメーションブロックの 18，19 バイト目を使っている。

ファイルの本体は無条件に G-RAM の 4000$_H$（青画面）からにロードされるので，ボーレートが目で見えるのである。48K バイト（C000$_H$バイト）を超える場合は，メイン RAM の BF00$_H$～FEFF$_H$にロードされる。なぜそのように中途半端なアドレスなのかというと，X1 の HuBASIC では FF00$_H$～を勝手に使えないからなのである。このことを一番手軽に確認するには，モニタに入ってダンプコマンドを実行してみればよい。しっかりキー入力バッファに使われてしまっているのである。てなわけで，使用に当たってはご注意である。

次は 580 行からのファイルの書き込みであるが，これは似たよーなものであるから割愛してしまって，さっさとリスト 13-1 の説明に行ってしまう。

まずは「PATRMM」というやつを見ていただきたい。察しのいい人なら分かるよーに，これはパッチ用のデータである。すなわち，このプログラムは自分自身を書き換えるのである。**嫌いなテクニック**を使ってしまって最高に気分が悪い私である。なんでこーなってしまったのかとゆーと，インフォメーションブロックやチェックサムはメイン RAM へリード（もしくはメイン RAM からライト）しなければならないが，データブロッ

クは無条件に G-RAM を対象にしているからである。フラグを使って分岐するとタイミングがその筋する可能性が高いのだ。ソフトウェアでタイミングを取っているプログラムの泣き所である。～MM というのがメイン RAM 用のパッチ，～IO というのが G-RAM 用のパッチである。

プログラムの実体は四つのジャンプから始まる。その次にパラメータエリアが並ぶ。では本筋の $BC27_H$ の「WHEAD1」を説明する。最初にやっているのは APSS 用の 8 秒間のブランクの書き込みである。8255 のビットセット/リセットコマンドを使っていることに注意。詳しくは第 0 章の 8255 ②の所を見ていただきたい。「WAITXX」というサブルーチンは，前述の「BREAK 信号」のチェックも行なうウエイトルーチンである。それが済んだ後にインフォメーションブロックを書き始めることになる。ソフトウェアでタイミングを取るプログラムであるから $BC33_H$ 番地の「DI」で割り込みを禁止していることに注目。次に，インフォメーションブロック用のデータはメイン RAM 上にあるので，$BC34_H$ ～ $BC39_H$ でパッチを行なっている。これによって「PATW」($BCBB_H$) からの 2 バイトが，

```
LD      (HL), D
NOP
```

に書き換わる。ここまできたなら，あとは HL レジスタにインフォメーションブロックのデータのアドレスを入れ (G-RAM の内容を書く場合は BC レジスタ)，DE レジスタにその長さを入れ，IY レジスタにマークの長さ ( 3 種類分) へのポインタを入れて「WRITGO」を CALL するわけである。

$BC4A_H$ 番地からの「WBODY1」も基本的には同じである。

最初にやっているのは GAP の書き込みである。インフォメーションブロックの直後にデータを書き込むと，タコなリードルーチンが読み飛ばしてしまうかもしれないので入れてある。本式にはどーするのか知らないが，世の中は丸く収まっているよーである。その後は G-RAM 方面へのパッチを行なって，各レジスタに値を放り込んで，「WRITGO」を CALL しているだけである。

以上のように書くと実に簡単そうに思えるかもしれないが，ここから先が七面倒なのである。

$BC6A_H$ からの「WRITGO」を見ていただきたい。3 回「WMARK」を呼んでマークを書いている。このサブルーチンはキャリーフラグがセットされているなら「 1 」を，リセットされているのなら「 0 」を，BC レジスタの数だけ書くのである。ただしこれは X1 フォーマットの場合である。例の $BC10_H$ 番地のフラグが 1 (MZ モード) のときは「 0 」,「 1 」が逆になる。$BCE4_H$ 番地あたりを見て納得していただきたい。その後はフォーマットどおりにスタートビットを書き，本体のバイト列を書き ($BC8E_H$ の「CALL WBYTES」)，最後にストップビットを書いている。

そこで問題点は $BC96_H$ 番地の「WBYTES」に移る。まず裏レジスタの HL を $0000_H$ にしている。これはチェックサム用のカウンタである。その後下働きの「WBS1」を呼び出している。これがバイト列ライトの直接の実行者である。このルーチンから帰ってきた後はまた「WBS1」を呼び出して 2 バイトのチェックサムを書いて終わりである。

「WBS1」($BCB5_H$) はスタートビットとして「1」を1回書いた後 ($BCCE_H$～$BCD1_H$), データを1バイトずつDレジスタに取り出して「WBYTE」を使って第7ビットから順に書いている（各バイトごとに各スタートビットも入れれば，1バイトごとに9ビット書いているわけだ)。「WBYTE」はチェックサム用に，$BCDA_H$番地で「1」を数えている。

ここでやっと「WBIT」($BCFB_H$)の説明に達した。このルーチンはキャリーフラグがセットされていれば「1」を，リセットされていれば「0」を書くわけである。$BCFC_H$から$BD11_H$までの間でその判定をして，A，A′レジスタにウエイト用の値を入れている。$BD10_H$の無駄な「JR」は，「1」と「0」での時間差をなくするためのもので縁起ものなので入れてある。念のために言っておくが,「1」は別名を「LONG」,「0」は別名を「SHORT」とも言うのである。つまり波形の長さからきた呼び方である。

「1」,「0」の書き方は少し考えれば分かるように，たとえばX1で「1」を書く場合は，I/Oの$1A02_H$番地の第0ビットをセット (High) して250 μsの間待ち，その後そのビットをリセット (Low) して250 μs待つのである（プログラムでは8255のビットセット/リセットコマンドを使っている)。

実を言うと，X1の場合はリセット (Low) しておく時間は少々短くしてある。理由は，このサブルーチンからリターンして**次の1ビットを書き始めるまでの間**もLowを書き続けているから，このサブルーチンの中でまるまる250 μs待っていると，Lowの期間が長くなりすぎてしまうからなのだ。この点に気を付けないと，X1ではリードエラーがバシバシ起きてしまうようである。しかしMZの方はこの点に関してはのんびりしているらしく，Lowが長すぎても問題はないようである。

$BD25_H$～$BD2D_H$は「BREAK信号」のセンスである。この分の時間もLowの長さに入れてやらなければならないので，(LONGL) と (SHORTL) の値はその分小さくなっている。あ～めんどくさ。

以上でテープライトルーチンの説明を終わるが，なぜかMZフォーマットは2回セーブのはずなのにそうなっていない。これを手抜きというのである。それはともかく，ここまで読んで気が付くように，このプログラムは**あんまり面白くない**（作った本人が言っているのだから本当である)。ほとんど唯一の特長は，わりかしきっちりしたトップダウンになっていることぐらいである。この事態を招いたもとは，テープのリード/ライトルーチンというものは，決まりきったことを決まりきったようにやるものだからということであろう。「可能な限りに短く作る」などの目標があればまた別なのだろーが。

と，グチをこぼしながらリードルーチンの説明に入る。$BD30_H$からである。「RHEAD1」,「RBODY1」の構造は基本的にライトルーチンと同じである。最初に**マーク**を検出するのであるが，ここでは1番目のマークははなから無視している。理由は，「2，3番目のマークだけで充分だから」と「MZではテープのリーダー部分のせいで，1番目のマークは決まった個数にならないから」である。繰り返すが，世の中は丸く収まっているのである。

というわけで，2番，3番目のマークを目標にして**同期を取って**実際のデータが書かれ

ている所に迫っていくのが $BD62_H$～$BD89_H$である。ここまできたなら，「RBYTES」を呼び出して，指定したバイト数を読んでしまう。その後はチェックサムを確かめて，合っていればOK，さもなければエラーである。

「RBYTES」，「RBS1」，「RBYTE」に関しては別に説明する必要はないだろう，ということで $BE14_H$番地からの「RBIT」である。

まず，$BE18_H$～$BE1F_H$のループである。ここは「テープから読んだ信号が High である限りループせよ」＝「Low になったら出ていいよ」ということである（BREAK 信号のチェックは別だよ）。ここを抜けてきたならば，そのときテープの信号は Low になっている。次にある $BE20_H$～$BE29_H$のループは「テープから読んだ信号が Low である限りループせよ」＝「High になったら出ていいよ」である。これら二つのループのやっていることは，

**エッジの検出**

なわけである。すなわち，テープの信号が Low → High になった直後には $BE2A_H$番地に来ているのである。そこで，（X1 の場合は）185 μs 待ってからもう一度テープの信号を読み取り，「1」なのか「0」なのかを見分けるのである。これがテープリードである。

プログラムでは $BE37_H$番地から，

```
LD     A, (AFTERW)
CALL  C, WAITX
```

などというものがある。これは，「1」を読んだ後は，（X1 の場合なら）次の 1 ビットまで 500－185＝315 μs も時間があり，へたをするとよけいなノイズを読んでしまうかもしれないので厄除けのつもりで入れたのだが，はっきり言ってほとんど役に立っていないよーである。6 バイト損こいた。なお，ライトルーチンと同じよーに，MZ フォーマットを読むときも二度読みはしていない。

そのよーなわけでプログラムの説明は一段落するのであるが，ここで一般教養としてソフトウェアでタイミングを取る場合に必要な「時間の計算方法」を示しておく。

てきとーな機械語の参考書を見ると，各命令ごとに「T（ステート数）＝××」という数字が書いてあるはずである。これが，「その命令を実行するのに何クロック必要か」ということである。たとえば，「LD　A，n」ならば T＝7 であるから，7 クロックかかるのである。そいでもって，X1 のクロックは 4MHz＝4000000Hz だから，1 クロックの時間は，

1 ÷4000000（Hz）＝0.00000025（秒）

となる。すなわち 1 クロック＝250 ns＝0.25 μs である。このことから，「LD　A，n」の実行時間は 0.25(μs)×7＝1.75(μs)と計算できる。実用的には，**すべてのステート数を足し合わせた数字を 4 で割り，それに「μs」を付ければよい。**

念のために書いておくと，

1ms（ミリ秒）　　＝0.001 秒
1μs（マイクロ秒）＝0.000001 秒
1ns（ナノ秒）　　＝0.000000001 秒

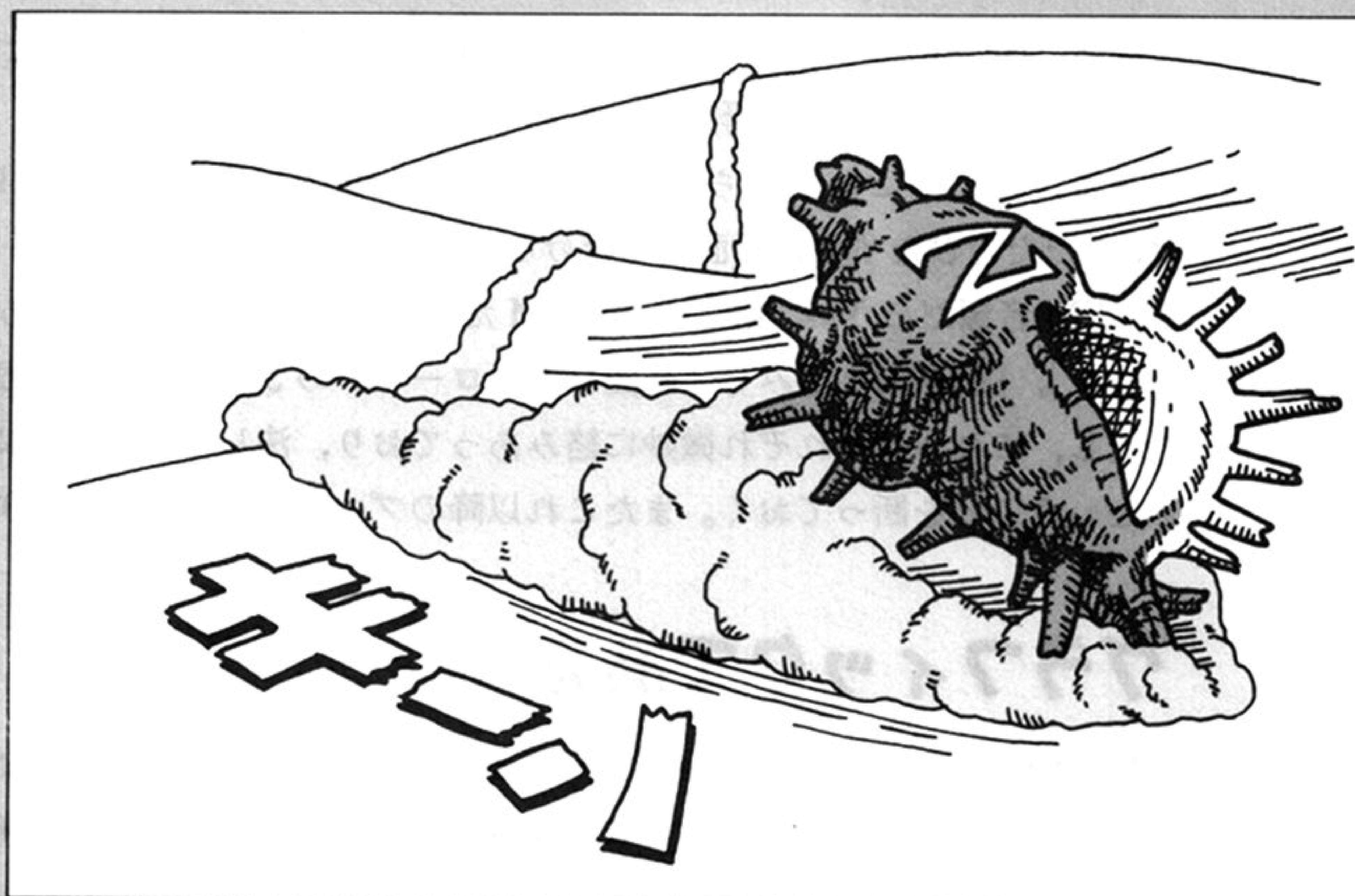
である。

そのよーなわけでテープであった。本章のサンプルプログラムは，このままでは単なるBASICテープをバックアップするぐらいのことしかできない。しかし頑張れば「テープ→フロッピーディスク転送プログラム」ぐらいにはなるだろう。また，G-RAMにロードされたプログラムをメインRAMに転送して実行する機械語ルーチンなどもおいしそうである。

最後に言っておくが，このプログラムは**プロテクトのかかったゲームプログラムなどは読めない。**読めないったら読めない。まかり間違っても「某ゲームのテープ版を読めないんですけど，バグじゃないですか」などという文句をつけてこないよーに。ボーレートを自由に変えれる所まではやったのだから，その先の**「不毛なあの筋」は自由研究である。**

# 第14章

# turbo Z




Zの機能はおいしいのである

# 第14章
# Zの機能はおいしいのである・・・・

この章ではturboZの機能についてやるのである。さっさと書いてしまうと，turbo（turboIIも含む）とturboZの差は，

1）アナログRGBにも対応し4096色を使用可能
2）各種モード付きの画像取り込み機能
3）第2水準ROMを標準搭載
4）2HDドライブ搭載
5）グラフィック2画面をプライオリティ付きで重ね合わせ表示可能
6）テキストパレットを使用可能
7）FM音源を標準で搭載
8）マウスを標準装備
9）スクロールイン/アウト機能
10）インタレーススーパーインポーズ
11）クロマキー（画面合成）可能

などとなっている。

なお，turboとturboZの間にはturboIIIという**微妙に緊張した関係**のマシンもあるが，それについては自由研究としておく。

さて，以上のことをつらつら考えるに，turboZにあってturboにない機能というのは，ただただひたすらに画面周りだけなのである。付けようと思えば2HDドライブだろうが，FM音源だろうが，第2水準ROMだろうが付いてしまうのである。そこで，「グラフィック」，「画像取り込み」，「画面コントロール」の3点に絞って取りあげることにした。ただし，これらはそれぞれ微妙に絡みあっており，決してバラバラに独立したものではないということを断っておく。またこれ以降のプログラムはCZ-8FB02を使うこと。

## グラフィックである

CZ-8FB03を使えばちょいちょいと4096色が使えるのだが，昔ながらのCZ-8FB02だとなかなかそうはいかない。そこで**リスト14-1**である。これは4096色の色見本である。グラフィック画面は320×200であるから，全部で64000ドットなわけだ。それを4096の色数で公平に配給すると，1色当たり15.625ということになる。よってあれこれ考えて，1色当たり縦3ドット×横4ドットの12ドットを，さらに縦横64色ずつ並べることにした。その処理をしているのが250〜400行である。カラーコードの並び方は，画面左上から右にかけて順に0，1，2，3，4，……，63となり，一段下がって左から64，65，66……という，ごく当たり前の配列になっている。で，おそらくは読者の中にはどー

してこのボックスフル (LINE～BF) の結果が 4096 色の色見本を作るのか分からない人がいるかもしれない。ま，それはそれでいいのである。「どしてかなー？」と考え続けていれば，そのうち分かるときがくれば分かるであろう。もしも分からなかったとしても，それはそれで別に悩むこともない。

リスト 14-1　4096 色見本+Z用パレットルーチン

```
100 CLEAR &HEF00
110 MEM$(&HEF00,16)=HEXCHR$("DD E5 FE 03 3E 0D C0 78 FE 04 3E 05 D8 E1 01 C5")
120 MEM$(&HEF10,16)=HEXCHR$("1F C5 3E 80 ED 79 EB 7E 23 ED 67 0F 0F 0F 0F 4F")
130 MEM$(&HEF20,16)=HEXCHR$("7E 23 E6 F0 57 ED 67 06 10 ED 79 04 7E 23 B2 ED")
140 MEM$(&HEF30,16)=HEXCHR$("79 04 7E E6 0F B2 ED 79 C1 AF ED 79 C9 01 C5 1F")
150 MEM$(&HEF40,16)=HEXCHR$("C5 3E 80 ED 79 CD 4D EF C1 AF ED 79 C9 11 10 10")
160 MEM$(&HEF50,16)=HEXCHR$("AF 67 6F 4F AF 42 ED 79 04 ED 61 04 ED 69 19 C6")
170 MEM$(&HEF60,16)=HEXCHR$("11 30 F2 0C C8 AC 20 EC 67 2C 18 E9 00 00 00 00")
180 DEFUSR0=&HEF00
190 '
200 INIT:KLIST 0
210 OPTION SCREEN 0:WIDTH 40,25,0,1
220 OUT &H1FB0,&H80
230 GOSUB"CLS12"
240 'CALL &HEF3D 'XPINIT
250 FOR L=0 TO 5
260   D=4*(2^L)
270   IF L<4 THEN C=1 ELSE C=2
280   S=3-(L MOD 4):SCREEN S,S
290   FOR J=0 TO 255 STEP D*2
300     LINE(J+D,0)-(J+2*D-1,191),XOR,C,BF
310   NEXT
320 NEXT
330 FOR L=6 TO 11
340   D=3*(2^(L-6))
350   IF L<8 THEN C=2 ELSE C=4
360   S=3-(L MOD 4):SCREEN S,S
370   FOR J=0 TO 191 STEP D*2
380     LINE(0,J+D)-(255,J+2*D-1),XOR,C,BF
390   NEXT
400 NEXT
410 FOR I=0 TO 4095
420   FOR C1=0 TO 4095
430     C2=C1+I
440 '   GOSUB"XPALET"
450     A$=USR0(MKI$(C1)+MKI$(C2))
460  NEXT
470 NEXT
480 END
490 '
500 LABEL"CLS12"
510 SCREEN,0:CLS4
520 SCREEN,1:CLS4
530 SCREEN,2:CLS4
540 SCREEN,3:CLS4
550 INIT
560 RETURN
570 '
580 LABEL"XPALET"
590 OUT &H1FC5,&H80
600 GR=INT((C1 AND &HFFF)/&H10)
610 B=(C1 AND &HF)*16
620 B0=INT(C2 AND &HF)
630 R0=INT((C2 AND &HF0)/&H10)
640 G0=INT((C2 AND &HF00)/&H100)
650 OUT &H1000+GR,B+B0
660 OUT &H1100+GR,B+R0
670 OUT &H1200+GR,B+G0
680 OUT &H1FC5,&H0
690 RETURN
```

で，色見本だけではナニなので，410～470 行でパレットをいじっている。見れば分かると思うが，最初のループ（0回目）で「PALET △，△」=初期化，次の1回目のループで「PALET △，△+1」，さらに「PALET △，△+2」……をやっているわけだ。これは無限ループになっているので止めるには BREAK キーである。450 行の USR0 を取り去

って440行のGOSUB文を生かすと，パレット変更ルーチンが580行からのBASIC部分になって，とても遅くなるので一度(一度で充分)試していただきたい。USR0で実行している機械語サブルーチンのアセンブルリストは**リスト14-2**である。で，それをよく見るとなんか変なのが後ろにくっついているわけだ。そう，これはパレット初期化ルーチンなわけだ。リスト14-1の240行にこっそりとあるように，「CALL &HEF3D」で4096色のパレットを初期化する。$EF3D_H$〜$EF6B_H$は簡単にリロケータブルになるので（自分でやっていただきたい），結構便利だろうと思う。ちなみにこのルーチンのメイン部分である$EF4D_H$〜$EF6B_H$（計31バイト）はその筋にコンパクト設計となっている。かなり苦労して作ったからこれよりも小さく作ることはかなり難しいであろう。それよりも，きっと解読するだけでも一苦労だと思う。ちなみにコメントがほとんど付いてないが，これは**わざとそうしたのである。**けけけけけ。

**リスト14-2　Z用パレット設定ルーチンとパレット初期化ルーチン**

```
                                .Z80
                                .PHASE   0EF00H
                        ;
1000                    BLUE    EQU      1000H
1100                    RED     EQU      1100H
1200                    GRN     EQU      1200H
                        ;
EF00    DD E5           XPALET: PUSH     IX
EF02    FE 03                   CP       3
EF04    3E 0D                   LD       A,13      ;Type mismatch
EF06    C0                      RET      NZ
                        ;
EF07    78                      LD       A,B
EF08    FE 04                   CP       4
EF0A    3E 05                   LD       A,5       ;Illegal f call
EF0C    D8                      RET      C
                        ;
EF0D    E1                      POP      HL        ;DROP
EF0E    01 1FC5                 LD       BC,1FC5H
EF11    C5                      PUSH     BC
EF12    3E 80                   LD       A,80H
EF14    ED 79                   OUT      (C),A     ;WRITE MODE
                        ;
EF16    EB                      EX       DE,HL
EF17    7E                      LD       A,(HL)
EF18    23                      INC      HL
                        ;(HL)=?.G,A=R.B
EF19    ED 67                   RRD                ;A=R.G,(HL)=B.?
EF1B    0F                      RRCA
EF1C    0F                      RRCA
EF1D    0F                      RRCA
EF1E    0F                      RRCA               ;A=G.R
EF1F    4F                      LD       C,A       ;C=G.G!!
EF20    7E                      LD       A,(HL)
EF21    23                      INC      HL
EF22    E6 F0                   AND      0F0H
EF24    57                      LD       D,A       ;D=B.0!!
                        ;
                        ;A=B.0,(HL)=r.b
EF25    ED 67                   RRD                ;A=B.b,(HL)=0.r
EF27    06 10                   LD       B,HIGH(BLUE)     ;=10H
EF29    ED 79                   OUT      (C),A
EF2B    04                      INC      B         ;B=11H
EF2C    7E                      LD       A,(HL)
EF2D    23                      INC      HL
EF2E    B2                      OR       D         ;A=B.r
EF2F    ED 79                   OUT      (C),A
EF31    04                      INC      B
EF32    7E                      LD       A,(HL)    ;A=?.g
EF33    E6 0F                   AND      0FH
EF35    B2                      OR       D         ;A=B.g
EF36    ED 79                   OUT      (C),A
EF38    C1                      POP      BC
EF39    AF                      XOR      A
```

```
EF3A    ED 79               OUT     (C),A    ;ACCESS OFF
EF3C    C9                  RET
                    ;
EF3D    01 1FC5     XPINIT: LD      BC,1FC5H
EF40    C5                  PUSH    BC
EF41    3E 80               LD      A,80H
EF43    ED 79               OUT     (C),A
EF45    CD EF4D             CALL    XPMAIN
EF48    C1                  POP     BC
EF49    AF                  XOR     A
EF4A    ED 79               OUT     (C),A
EF4C    C9                  RET
                    ;
EF4D    11 1010     XPMAIN: LD      DE,1010H
EF50    AF                  XOR     A
EF51    67                  LD      H,A
EF52    6F                  LD      L,A
EF53    4F                  LD      C,A
EF54    AF          L1:     XOR     A
EF55    42          L2:     LD      B,D      ;D=10H
EF56    ED 79               OUT     (C),A
EF58    04                  INC     B
EF59    ED 61               OUT     (C),H
EF5B    04                  INC     B
EF5C    ED 69               OUT     (C),L
                    ;
EF5E    19                  ADD     HL,DE
EF5F    C6 11               ADD     A,11H
EF61    30 F2               JR      NC,L2
                    ;
EF63    0C                  INC     C
EF64    C8                  RET     Z
                    ;
EF65    AC                  XOR     H
EF66    20 EC               JR      NZ,L1
EF68    67                  LD      H,A
EF69    2C                  INC     L
EF6A    18 E9               JR      L2
                    ;
                            END
```

では説明に入る。そこでリスト14-1の580～690行なわけだ。これがパレット設定の基礎である。

ここでは「PALET C1, C2」に相当することをしているわけである。まず大事なことはI/Oの$1FC5_H$のビット7，ビット3である。これは第0章を見れば分かるであろうが，turboZモードでアナログRGBのパレット設定に先立っては，「ビット7＝1，ビット3＝0」としておかなければならないのである。ビット7＝1はアクセスON，ビット3＝0は書き込みモード（1なら読み出しモード）ということである。そして，できれば680行にもあるように，設定が済んだ後は「ビット7＝0」としてアクセスをOFFにしておくべきである。さもないと，LIST, INIT命令などの「パレットをいじるが，アナログRGBには対応していない命令」が，おかしなパレット設定をしてしまうことがある。

その後は，3回のOUT命令を実行するのだが，その方法がちょっと違和感のある形式になっている。が，一つ例をあげればたちまち分かってしまうだろう。すなわち，

PALET　&H○△□,　&H$\alpha\beta\gamma$

は，

OUT　&H10○△,　&H□$\gamma$
OUT　&H11○△,　&H□$\beta$
OUT　&H12○△,　&H□$\alpha$

になるのだ。ただし○△□，$\alpha\beta\gamma$ というのは，それぞれ「1桁の16進数」である。具体的な数字をあげると，

PALET　&HABC，&HDEF

は，

OUT　&H10AB，&HCF

OUT　&H11AB，&HCE

OUT　&H12AB，&HCD

となるのだ。「C」が「OUTされる数値」の方に回されているというのが微妙なのである。ま，これだけ言っておけば大丈夫であろう。あとはリスト14-2であるが，これは「RRD」の使い方がなかなかである。この命令は「Aレジスタの下位4ビット」，「(HL)の上位4ビット」，「(HL)の下位4ビット」の三つの間で値を回すという命令である。詳しくは参考文献4を見ていただきたい。で，気を付けなければいけないのが，この命令を実行すると(HL)が書き換わるということである。よって，たとえばリスト14-1の450行を，

X$＝MKI$(C1)＋MKI$(C2)：A$＝USR0(X$)

などとすると，X$の中身が書き換わってしまうのである。というわけで，この点は注意するよーに。

次に**リスト14-3**である(100〜200行と480行以降はリスト14-1と同じである)。これは64色×2画面モードである。280〜370行で64色モード用のパレットを設定し，390〜470行で二つのグラフィック画面＋キャラクタのプライオリティ(優先順位)をパコパコと切り換えている(ついでにテキストパレットもランダムに変えている)。何かキーを押すと終了する。そいでもって，このときのパレットの設定であるが，「R，G，Bそれぞれの下位2ビットが無効」ということになっている。すなわち，

| | | | | | | |
|---|---|---|---|---|---|---|
| 青→0001 | 0000 | ＊＊×× | ＊＊×× | ： | ＊＊×× | ＊＊＊＊ |
| 赤→0001 | 0001 | ＊＊×× | ＊＊×× | ： | ＊＊×× | ＊＊＊＊ |
| 緑→0001 | 0010 | ＊＊×× | ＊＊×× | ： | ＊＊×× | ＊＊＊＊ |
| $10_H$，$11_H$，$12_H$ | | G | R | | B | G′，R′，B′ |

において，「＊」が有効なビットで，「×」が無効なビットということである。「×」の部分は注意深く避けないと，予期せぬパレットが設定されたりする。なぜかというと，たとえば「OUT &H10FF, &HF？」と「OUT &H10CC, &HC？」は同じだからなのだ(無効ビットを0として考えれば分かる)。よって，前者と後者の両方を実行して，なおかつ「？」の部分が双方で違っていたなら，「あれ？　パレットがヘンだぞ」ということになるのである。ま，ここらへんは結構複雑で面倒臭いから，しばらくあれこれといじってみるのが一番手っ取り早いであろう。

**リスト14-3　64色×2　サンプル**

```
100 CLEAR &HEF00
110 MEM$(&HEF00,16)=HEXCHR$("DD E5 FE 03 3E 0D C0 78 FE 04 3E 05 D8 E1 01 C5")
120 MEM$(&HEF10,16)=HEXCHR$("1F C5 3E 80 ED 79 EB 7E 23 ED 67 0F 0F 0F 0F 4F")
```

```
130 MEM$(&HEF20,16)=HEXCHR$("7E 23 E6 F0 57 ED 67 06 10 ED 79 04 7E 23 B2 ED")
140 MEM$(&HEF30,16)=HEXCHR$("79 04 7E E6 0F B2 ED 79 C1 AF ED 79 C9 01 C5 1F")
150 MEM$(&HEF40,16)=HEXCHR$("C5 3E 80 ED 79 CD 4D EF C1 AF ED 79 C9 11 10 10")
160 MEM$(&HEF50,16)=HEXCHR$("AF 67 6F 4F AF 42 ED 79 04 ED 61 04 ED 69 19 C6")
170 MEM$(&HEF60,16)=HEXCHR$("11 30 F2 0C C8 AC 20 EC 67 2C 18 E9 00 00 00 00")
180 DEFUSR0=&HEF00
190 '
200 INIT:KLIST 0
210 OPTION SCREEN 0:WIDTH 40,25,0,1
220 OUT &H1FB0,&H80
230 GOSUB"CLS12"
240 CALL &HEF3D 'XPINIT
250 SCREEN,0:SYMBOL (0,0),"0",8,4,1,0
260 SCREEN,2:SYMBOL (30,30),"2",8,4,2,0
270 '
280 OUT &H1FB0,&H90
290 FOR G=0 TO 15 STEP 4
300   FOR R=0 TO 15 STEP 4
310     FOR B=0 TO 15 STEP 4
320       C1=G*&H100+R*&H10+B
330       C2=C1
340       A$=USR0(MKI$(C1)+MKI$(C2))
350     NEXT
360   NEXT
370 NEXT
380 '
390 SCREEN 0,0:LOCATE 0,5:PRINT STRING$(12,"@")
400 FOR J=0 TO 8 STEP 8
410   FOR I=0 TO 2
420     D=16+J+I:LOCATE 20,15:PRINT HEX$(D)
430     OUT &H1FC0,D:PAUSE 15
440   NEXT
450 NEXT
460 IF INKEY$="" THEN 400
470 END
480 '
490 LABEL"CLS12"
500 SCREEN,0:CLS4
510 SCREEN,1:CLS4
520 SCREEN,2:CLS4
530 SCREEN,3:CLS4
540 INIT
550 RETURN
```

turboZ では上記のプログラムで示した以外にもさまざまな画面モードがあるわけだ。それらの場合にはどんな色数が使えて，パレットの設定はどうするかという問題がある。それにいちいち答えるのも面倒なので，一番おいしそうな「640×400 で 4096 色中から 8 色を使用可能」を**リスト 14-4** に示す。

**リスト 14-4　640×400 で 4096色中 8 色モード**

```
100 CLEAR &HEF00
110 MEM$(&HEF00,16)=HEXCHR$("DD E5 FE 03 3E 0D C0 78 FE 04 3E 05 D8 E1 01 C5")
120 MEM$(&HEF10,16)=HEXCHR$("1F C5 3E 80 ED 79 EB 7E 23 ED 67 0F 0F 0F 0F 4F")
130 MEM$(&HEF20,16)=HEXCHR$("7E 23 E6 F0 57 ED 67 06 10 ED 79 04 7E 23 B2 ED")
140 MEM$(&HEF30,16)=HEXCHR$("79 04 7E E6 0F B2 ED 79 C1 AF ED 79 C9 01 C5 1F")
150 MEM$(&HEF40,16)=HEXCHR$("C5 3E 80 ED 79 CD 4D EF C1 AF ED 79 C9 11 10 10")
160 MEM$(&HEF50,16)=HEXCHR$("AF 67 6F 4F AF 42 ED 79 04 ED 61 04 ED 69 19 C6")
170 MEM$(&HEF60,16)=HEXCHR$("11 30 F2 0C C8 AC 20 EC 67 2C 18 E9 00 00 00 00")
180 DEFUSR0=&HEF00
190 '
200 INIT:KLIST 0
210 OUT &H1FB0,&H80
220 OPTION SCREEN 0:WIDTH 80,25,1,2
230 CLS4
240 '
250 FOR I=0 TO 7
260   C1=((I ¥ 4) AND 1)*&H800+((I ¥ 2) AND 1)*&H80 +(I AND 1)*&H8
270   C2=((I ¥ 4) AND 1)*&HF00+((I ¥ 2) AND 1)*&HF0 +(I AND 1)*&HF
280   A$=USR0(MKI$(C1)+MKI$(C2))
290 NEXT
300 '
310 FOR I=0 TO 7
320   LINE(I*30,I*20)-(100+I*30,100+I*20),PSET,I,BF
330 NEXT
```

```
340 '
350 FOR I=0 TO 7
360   C1=((I ¥ 4) AND 1)*&H800+((I ¥ 2) AND 1)*&H80 +(I AND 1)*&H8
370   C2=RND(1)*4096
380   A$=USR0(MKI$(C1)+MKI$(C2))
390 NEXT
400 GOTO 350
```

で，これらの基本は，

1） 低解像度（15kHz）のときは，多色モードにするとバンク０，バンク１が重ね合わさって一つの色になる

2） 高解像度（24kHz）のときは，多色モードにするとバンク０，バンク１が別々になり，同時に両方の面を見ることができない。それに伴い，$1FC0_H$番地の優先順位などは意味を持たなくなる（キャラクタとの間の優先順位はあるが）

3） 色数は，320×200の4096色モード以外は，640×400のときが８色なだけで後はすべて１画面64色である

4） パレットの指定は常にRGBの下位ビットから無効になっていく

なお補足しておくが，ZではIN命令により，その時点でのパレット設定値の読み出しも可能になっている。X1/turboでは値を読み出せなかっただけではなく，パレットの設定が変わってしまったりしていたのである。これは特に大きなメリットがあるわけでもないが（もちろんCZ-8FB03のPSAVEなどはおいしいが），とりあえずは前進であろう。具体的な，パレット設定値の読み出し方法は，

1） $1FC5_H$の第７ビットと第４ビットを１にする。これで多色モードかつ，パレットは読み出しモードとなる

2） OUT &H10GR，B？を実行する。「？」は何でもよい。このとき，パレットは読み出しモードなので，新しい値の設定はされないことに注意。この後，B′＝INP（&H10GR）AND &H0Fとする

3） 「&H10」を「&H11」に置き換えてR′を得る

4） 「&H10」を「&H12」に置き換えてG′を得る

5） こうして得た&HG′R′B′がパレットの設定値

ということになる。

ところで，テキストパレットはそれぞれに64色（階調）を指定できるのであるが，美しくないことにカラー０＝黒は黒のまんまなのである。I/Oアドレスから見ると$1FB8_H$番地が臭いのであるが，残念ながらだめである。すなわち，この機能を使ってキャラクタの黒色を他の色に変えることはできないよーである。

で，極めて当然のことながら，テキストパレットはPCGにもかかわってくるのである。よって，turboZでは，PCGで使える色数が大盤振舞されているのである。

## 画像取り込み

これはなかなか画期的な機能であるが，使い方はものすごく簡単なのでプログラムだけ載せておく。まず**リスト 14-5**は単純に画像取り込みをしつつ，位置補正をするプログラム

である。110 行の所でマイナスの値を入力すると取り込みをやめるのである。ちなみに，CRT3 というのはスーパーインポーズをする命令である。turboZ で画像取り込みをするためには，スーパーインポーズ画面でなくてはいけないのだ。

**リスト 14-5　画像取り込み＋位置補正**

```
100 WIDTH40,25,0,1:OUT&H1FB0,&H88:CRT 3
110 INPUT A
120 IF A>=0 THEN OUT &H1FC1,A:GOTO 110
130 OUT &H1FB0,&H80:CRT 1
140 END
```

次に**リスト 14-6** である。これはモザイク処理である。入力は 8 進数で行なうことと，X，Y 方向のモザイク指定には無効な部分があることに注意。ちなみに「56」を入力して大きなモザイクにした後で SHIFT ＋ . でコンピュータ画面にすると，それはそれなりに楽しい画面となる。入力のときに 8 進数ではなく「 .. 」を入力するとクロマキーのデモとなる。これはさりとて面白いわけではない。無限ループだから適当な所で BREAK すること。

**リスト 14-6　モザイク＋クロマキー　サンプル**

```
100 WIDTH40,25,0,1:OUT&H1FB0,&H88:CRT 3
110 OUT &H1FC2,0
120 OUT &H1FC3,0
130 OUT &H1FC1,40
140 INPUT A$:IF A$=".." THEN 180
150 A=VAL("&O"+A$)
160 OUT &H1FC2,A:GOTO 140
170 '
180 OUT &H1FC2,0
190 OUT&H1FB0,&H80:CRT 3
200 FOR X=&H80 TO &HC0 STEP &H40
210   FOR I=0 TO 7
220     C=(I AND 4)*8+(I AND 2)*4+(I AND 1)*2
230  PRINTBIN$(X OR C)
240     BEEP:OUT &H1FC3,X OR C:PAUSE 10
250   NEXT
260 NEXT
270 GOTO 200
```

そして最後が**リスト 14-7** のスクロールである。turboZ にはスクロール IN/OUT という機能が付いているが，それのサンプルである。お勧めの入力は 8，9，10，11，12 などである。そのうち「12」は，CRT 出力をカットすることになる。これは別にスクロールと組み合わせる必要はなく，画像取り込み中であろうが，スーパーインポーズ中であろうが，普通のコンピュータ画面であろうが，とにかく CRT 出力をカットしてしまう。ビデオマニアにはおいしいであろう。ところでなかなか面白いのがスクロールとモザイク処理付きの画像取り込みの組み合わせである。モザイク処理部分がクルクルと回ってなかなかに面白い。

以上であるが，細かい点であちこちに取りこぼしもあるだろう。一つには turboZ の拡張機能は，カスタム LSI を使ったものであるために，面白そうなもの，便利そうなものを何

リスト14-7　スクロール　サンプル

```
100 WIDTH40,25,0,1:CRT 3
110 SCROLL 2
120 INPUT N:IF N<0 THEN 160
130 OUT &H1FC4,N
140 GOTO 120
150 '
160 SCROLL 0
170 OUT &H1FC4,0
```

でもかんでも寄せ集めたという感じだからでもある。たとえば画像取り込みに，「反転階調」などというものも指定できるのであるが，どう考えてもこれの用途は，反転をボカシの代わりにしたその筋のビデオを，もう一度その筋する機能ぐらいの意味しかないように思うのだが，どうであろうか（今度やってみよ）。それ以外にもやり残したのに「インタレーススーパーインポーズ」もあるが，これはビデオ編集専用であろう。その方面の趣味を持っている方はスクロールIN/OUTの機能とともに大変おいしいであろう。てなところでturboZであった。

# 付　録

## A　ダンプリストチェック用プログラム

## B　X1処理技術者試験

## C　初出一覧・参考文献

# 付録A ダンプリストチェック用プログラム

リストA-1はダンプリストチェック用のプログラムである。ごちゃごちゃしているので，打ち込みには充分に注意すること。リストA-2が機械語部分のアセンブルリスト，リストA-3がその部分を自分自身で表示させたものである。

**リストA-1** ダンプリストチェックツール

```
100 T=&HC000:CLEAR T
110 MEM$(T+ 0,16)=HEXCHR$("05 C8 EB E5 56 23 5E 23 05 28 1C EB 0E 80 1A A1")
120 MEM$(T+16,16)=HEXCHR$("28 01 37 ED 6A 30 08 3E 10 AC 67 3E 21 AD 6F CB")
130 MEM$(T+32,12)=HEXCHR$("09 30 EB 13 10 E8 EB E1 73 23 72 C9")
140 DEFUSR0=T
150 DEFINT B-Z:DIM X(7)
160 INPUT A$
170 B=VAL("&H"+MID$(A$,2,4)):IF B>=0 THEN 160
180 C=VAL("&H"+MID$(A$,7,4)):IF C>=0 THEN 160
190 FOR D=B TO C STEP 128
200 IF D+127<C THEN E=127 ELSE E=C-D
210 Z$=LEFT$(MEM$(D,E+1)+STRING$(127,0),128)
220 F=0:FOR J=0 TO 7:X(J)=0:NEXT
230 FOR I=0 TO 15
240 IF D+F<=C THEN E$=HEX$(D+F)+" ":GOSUB 360 ELSE I=99:GOTO 290
250 Y=0:FOR J=0 TO 7
260 G=ASC(MID$(Z$,F+1,1)):Y=Y+G:X(J)=X(J)+G
270 IF F<=E THEN E$=RIGHT$("0"+HEX$(G),2)+" " ELSE I=99:E$=SPACE$(3)
280 GOSUB 360:F=F+1:NEXT:E$=": "+RIGHT$("0"+HEX$(Y),2):GOSUB 340
290 NEXT:E$=STRING$(33,"-"):GOSUB 340:E$="SUM: ":GOSUB 360
300 FOR J=0 TO 7:E$=RIGHT$("0"+HEX$(X(J)),2)+" ":GOSUB 360:NEXT
310 Z$=USR0(LEFT$(Z$,E+1)):IF E=0 THEN Z$=Z$+Z$
320 E$=RIGHT$("000"+HEX$(CVI(LEFT$(Z$,2))),4):GOSUB 340:E$="":GOSUB 340
330 NEXT:END
340 GOSUB 360:IF LEFT$(A$,1)="P" THEN LPRINT ELSE PRINT
350 RETURN
360 IF LEFT$(A$,1)="P" THEN LPRINT E$; ELSE PRINT E$;
370 RETURN
```

**リストA-2** CRC計算プログラム

```
                                   .Z80
                                   .PHASE  0C000H   ;OR ANY PLACE
                           ;
                           ;DE=TOP ADDR
                           ;B=LENGTH
C000    05                 MKCRCS: DEC     B
C001    C8                         RET     Z        ;LEN=1 THEN RET
                           ;
C002    EB                         EX      DE,HL
C003    E5                         PUSH    HL       ;SAVE VAL ADDR.
C004    56                         LD      D,(HL)
C005    23                         INC     HL
C006    5E                         LD      E,(HL)
C007    23                         INC     HL
                           ;
C008    05                         DEC     B
C009    28 1C                      JR      Z,FINIS  ;JUST 2
                           ;
C00B    EB                         EX      DE,HL    ;HL=SEED,DE=ADDR.
C00C    0E 80                      LD      C,80H    ;MASK
                           ;
C00E    1A                 LOOP:   LD      A,(DE)   ;DATA
C00F    A1                         AND     C        ;CHECK
C010    28 01                      JR      Z,SKIP
                           ;
C012    37                         SCF              ;CY=1
C013    ED 6A              SKIP:   ADC     HL,HL    ;HL=HL+HL+CY
C015    30 08                      JR      NC,NEXT  ;NEXT BIT
```

```
                                ;
C017    3E 10                           LD      A,10H
C019    AC                              XOR     H
C01A    67                              LD      H,A
C01B    3E 21                           LD      A,21H
C01D    AD                              XOR     L
C01E    6F                              LD      L,A       ;HL=HL XOR 1021H
                                ;
C01F    CB 09                   NEXT:   RRC     C         ;ROTATE MASK
C021    30 EB                           JR      NC,LOOP   ;RIGHT BIT
C023    13                              INC     DE        ;NEXT BYTE
C024    10 E8                           DJNZ    LOOP      ;DEC COUNT
                                ;
C026    EB                              EX      DE,HL
C027    E1                      FINIS:  POP     HL        ;GET ADDR.
C028    73                              LD      (HL),E
C029    23                              INC     HL
C02A    72                              LD      (HL),D    ;SAVE
                                ;
C02B    C9                              RET
                                ;
                                        END
```

**リスト A-3** 出力サンプル

```
C000 05 C8 EB E5 56 23 5E 23 : 97
C008 05 28 1C EB 0E 80 1A A1 : 7D
C010 28 01 37 ED 6A 30 08 3E : 2D
C018 10 AC 67 3E 21 AD 6F CB : 69
C020 09 30 EB 13 10 E8 EB E1 : FB
C028 73 23 72 C9             : D1
---------------------------------
SUM: BE F0 02 D7 FF 68 DA AE C076
```

使い方を説明する。まずはリストA-1を間違いないように打ち込む。特に110行～130行は注意して打ち込むこと。打ち込み終わったら**RUNする前にセーブしておく**。そして，そのディスク（もしくはテープ）を取り出しておく。その後にRUNして動作チェックを行なう。RUNした後，「DC000 C02B⏎」でリストA-3のように表示したらひとまずはOKである（あくまで**ひとまず**である。まだまだバグが入っている可能性がある）。打ち込み間違いがないということはまずあり得ないから，一発で動くとは期待しないように。バグを取っている間，暴走させてディスクを壊したりしないように，くれぐれも注意していただきたい。

で，プリンタに打ち出す場合は「PC000 C02B」のように「P開始アドレス　終了アドレス」である。**アドレスは常に4桁の16進数で指定すること**。またこのプログラムでは0000～7FFFのアドレスは指定できないようになっている（必要もない）。

チェックするポイントは，リストA-3で　　　　になっている部分である。特に右下隅の4桁の16進数は大事である。

また，このプログラムで第10章のMMLプログラムをチェックするときは，前もってNEW ON &HD000を実行しておいてからリストA-1とMMLプログラムをロードし，100行の「CLEAR T」を削除してからRUNしていただきたい。

余談であるが，100行のT＝&HC000は，別にC000でなくてもよく，D000，E000，F000などでもかまわない。よーするに**チェックする機械語プログラムと重ならないアドレスならどこでもよい**のである。

# 付録B X1処理技術者試験

最近は軟弱なパソコンユーザーが増えている。

そんなことではいけないので，ここにとりあえずX1処理試験を用意することにした。おのおのの段階に応じて，第2種その筋，第1種その筋，特種その筋が認定される。ただし面倒臭いので，勝手に採点して勝手に認定されていただきたい。

なお，試験問題にはロクでもないものも含まれているので，心してかかるように。

**注意事項**

試験時間は無制限の金網デスマッチ3本勝負である。リターンマッチも可であるから，大いに頑張っていただきたい。ただし，ずるはしないように。

## 1．適性試験

**問** 以下の項目にYES/NOで答えよ

(1) 必要ないと分かっていても，ついつい最適化してしまう
(2) どんなプログラムでも「もう1バイト短くできる」と信じている
(3) どんなプログラムでも「もう1クロック速くできる」と信じている
(4) 指を10ビットのメモリとして使ったことがある
(5) プログラミングしながら日の出を迎えることは，それほど珍しくない
(6) CRTは目に悪いと体感している
(7) 戦闘機のフライトシミュレータは面白そうなゲームである
(8) 10進法は不自然である
(9) 頭の中に裏レジスタがある
(10) 「CD」とは音楽用語ではなくCALLのことである
(11) 「地震　雷　火事　停電」である
(12) 頭の中にフラグレジスタがある
(13) ディスケットの磁性面に触るようなやつは死刑にすべきだ
(14) パソコンの前で食事することが多い
(15) プリンタ用紙にメモすることが多い
(16) BASICよりも機械語の方が便利だ
(17) 老後の趣味はプログラミングと決めている

以上17項目のうち9項目以上にYESと答えた方はその筋陽性ですので，第2種その筋を名乗ることを許可します。

## 2．第1種その筋試験

**問1** PSGのレジスタ設定について答えよ

a) 花子さんはおかあさんからX1のPSGを使って300Hzの音を出して欲しいとたのまれました。チャンネルAを使うとしたら，$R_0$，$R_1$にはどんな数を設定すれば，一

番300Hzに近い周波数の音が出るでしょうか。

b)　太郎君はその筋なので，X1を使って出せる一番低い音（周波数の小さい音）を出してみようとしました。さて，太郎君が出した音の周波数はいくつでしょう。小数点第2位で四捨五入して答えなさい。

**問2**　パレットの設定について答えよ

NEW BASICとturbo BASICのコマンド「PALET@」は一つのコマンドで最大八つのパレットを設定できるコマンドである(知らなかったら×)。

そこで「PALET@文に相当する3バイトのデータ」について答えよ。

a)　資料を見ずに，初期状態の「3バイト」を答えよ。また，その3バイトはそれぞれ何番地にOUTするか？

b)　「PALET@　4，1，2，6，4，1，2，6」に対応する3バイトは？

c)　グラフィックを4色だけに制限する代わりに，青＞赤＞緑（＋黒）の優先順位で使えるようにしたい（勝手な話だが，この時点で何を言おうとしているのか分からなかったら×)。青をPLANE 0($4000_H$～$7FFF_H$)，赤をPLANE 1 ($8000_H$～$BFFF_H$)，緑をPLANE 2 ($C000_H$～$FFFF_H$)とするには，どうすればよいか。「PALET@　?，?，?，?，?，?，?，?」と「3バイトのデータ」で答えよ。

d)　c)と似たケースで，白(PLANE0)＞青(PLANE1)＞赤(PLANE2)（＋黒）の場合について答えよ。

e)　4色ではなく5色に制限して青，赤，緑＞白（＋黒）としたい。パレットコードを青＝1，赤＝2，緑＝3＞白＝4　（黒＝0）とするにはどうするか。「PALET@　?，?，?，?，?，?，?，?」と「3バイトのデータ」で答えよ。

**問3**　次の式をウンウン言いながら暗算して16進数で答えよ。ただし$\sim_B$は2進数，$\sim_O$は8進数を表すものである。

a)　$FA_H+27_H+34_H$

b)　$FF_H\times1011_B$

c)　$774_O+444_O$

d)　$10_H\times11_H+2A_H$

e)　$0F_H\times0F_H$

f)　$716_H\div2$

g)　$\sqrt{100_H}$

h)　$\sqrt{144_H}$

i)　$\sqrt{2710_H}$

**問4**　次のサブルーチンの実行時間を計算し，クロック数(ステート数)と時間(μsの単位)を答えよ。もちろんクロックは4MHzとする。ただしZ80の解説書を参照してもよろしい。

a)

```
        LD      A, 100
LOOP:   DEC     A
        RET     Z
        JR      LOOP
```

b)

```
        LD      A, 100
LOOP:   DEC     A
        JP      NZ, LOOP
        RET
```

c)

```
        LD      A, 100
LOOP:   DEC     A
        JR      NZ, LOOP
        RET
```

d)

```
        PUSH    AF
        PUSH    DE
        LD      DE, 151
LOOP:   DEC     DE
        LD      A, D
        OR      E
        JR      NZ, LOOP
        JR      SKIP
SKIP:   OR      A
        POP     DE
        POP     AF
        RET
```

**問5**　HLレジスタ×msだけ(よーするにHL＝1なら1ms，HL＝2なら2ms)時間つぶしをするサブルーチンを作りたい（HL＝0のときは考えなくてよい)。ただし，呼び出す側での，

```
PUSH  HL
LD    HL,    nm  ; 何m秒か
CALL  そのサブルーチン
```

　　POP　　HL
の時間も入れて考えるとする。

a)　空いてる所をその筋な文字列で埋めなさい（甘えは許しません）

```
        呼び出し例
            PUSH     HL        ;11
            LD       HL,1234   ;10
            CALL     WAIT      ;17
            POP      HL        ;10


        サブルーチン
WAIT:       PUSH     AF        ;11
            PUSH     DE        ;11
            NEG                ;8
            DEC      HL        ;6
            LD       A,H       ;4
            OR       L         ;4
;
            JR       (1),CASE0 ;12/7
            LD       DE,(2)    ;10
            CALL     WS        ;17
            BIT      0,(HL)    ;12
            DEC      DE        ;6
            OR       A         ;4
;
LOOP:       LD       DE,(3)    ;10
            NEG                ;8
            NEG                ;8
            CALL     WS        ;17
            DEC      (4)       ;6
            LD       (5)       ;4
            OR       (6)       ;4
            JR       NZ,LOOP   ;12/7
;
FINIS:      POP      DE        ;10
            POP      AF        ;10
            RET                ;10
;
CASE0:      LD       DE,(7)    ;10
            CALL     WS        ;17
            JR       FINIS     ;12
;
WS:         DEC      DE        ;6
            LD       A,D       ;4
            OR       L         ;4
            JR       NZ,WS     ;12/7
            RET                ;10
```

b)　上記のプログラムを短く作り変えなさい（これは簡単）

## 3．特種その筋試験（論文）

a)　第14章のリスト14-2（turboZの4096色パレットの初期化）の $EF4D_H$ ～ $EF6B_H$ の部分の動作を理解し解説せよ。

b)　それと同機能で，より短いサブルーチンを作れ。もしも不可能と考える場合は，不可能であることを証明せよ。

【解答】

第1種その筋試験

問1

a) $R_0$＝161，$R_1$＝1

b) 30.5Hz

問2

a) $AA_H$，$CC_H$，$F0_H$ の3バイト

OUTする番地は，

$AA_H$→$1000_H$番地

$CC_H$→$1100_H$番地

$F0_H$→$1200_H$番地

b) $22_H$，$CC_H$，$99_H$

c) PALET@　0，1，2，1，4，1，2，1

$AA_H$，$44_H$，$10_H$

d) PALET@　0，7，1，7，2，7，1，7

$EE_H$，$BA_H$，$AA_H$

e) PALET@　0，1，2，4，7，1，2，4

$32_H$，$54_H$，$98_H$

問3

a) $155_H$　b) $AF5_H$　c) $320_H$　d) $13A_H$　e) $E1_H$　f) $38B_H$　g) $10_H$
h) $12_H$　i) $64_H$

問4

a) 2695クロック，673.75$\mu$s

b) 1417クロック，354.25$\mu$s

c) 1612クロック，403$\mu$s

d) 3999クロック，999.75$\mu$s（けけけけけ）

問5

a)

(1) Z　(2) 147　(3) 151　(4) HL　(5) A，H　(6) L　(7) 147

b)

自己採点しなさい。②と⑦がともに「147」であることを使えば簡単。もちろんそれ以外でもよい。

特種その筋試験

解答が載っているなどと思っていた人は失格。

# 付録C 初出一覧・参考文献

**【初出一覧】**

各章のOh！MZ誌連載時における掲号とタイトルを下記に示す。

第0章　きっと完全無欠なI/Oマップ——'85年6月号「第1回 たぶん完全無欠なI/Oマップ」，'87年2月号「第21回 ほとんど完全無欠なI/Oマップ」

第1章　CRTCでどすこいである——'85年7月号「第2回 そこにCRTCがあるからなのだ」

第2章　PCGは二度おいしいのである——'86年3月号「第10回 PCGのお通りである」

第3章　漢字名野出亜留——'86年11月号「第18回 とーとー漢字なのである」

第4章　サブCUPのおかげなのである——'85年10月号「第5回 サブCUPは必修科目なのである」

第5章　CTCは律儀なのである——'86年4月号「第11回 CTCはきちょーめんなのである」

第6章　SIOでマウスである——'86年5月号「第12回 SIOは通信ばかりではないのである」

第7章　通信だってするのである——'87年8月号「最終回　通信プログラムである」

第8章　DMAはヘビー級である——'85年8月号「第3回 Z80の一族はただ者ではないのである」，'85年9月号「第4回 DMAはグラフィックもしてしまうのである」

第9章　ディスクを回すのである——'85年12月号「第7回 軟式円盤の基礎である」，'86年1月号「第8回 FDCは挑戦的である」，'86年2月号「第9回 FDCは業師なのである」，'86年7月号「第14回 DMAにはディスクがよく似合うのである」

第10章　PSGは基本である——'85年11月号「第6回 PSGはてりめぇである」

第11章　FM音源ナハトムジーク——'87年5月号「第23回 FM変調するのである」，'87年6月号「第24回 MMLを作るのである」，'87年7月号「第25回 MMLを完成するのである」

第12章　カラーイメージボードで取り込むのである——'87年4月号「第22回 カラーイメージボードなのである」

第13章　テープもやってしまうのである——'86年6月号「第13回 あげくのはてにはテープなのである」

**【参考文献】**

1．「X1マシン語活用百科」，清水保弘著，産業報知センター刊

2．X1 turbo 回路図公開，「Oh！MZ」，'84年4月号

3．続インターフェイスLSIの研究 CRTC［HD46505］編，「マイコンピュータNo.8」，千葉憲昭著，CQ出版社刊

4．「Z80ファミリ・ハンドブック」，額田忠之著，CQ出版社刊

5．「プログラマブルサウンドジェネレータ データマニュアル」，ゼネラル インスツルメント インターナショナル コーポレーション（GI）刊

6．「YM2151ユーザーズマニュアル」，日本楽器製造株式会社刊

7．「CZ-8BS1取扱説明書」，シャープ刊

8．「X1システム研究室」，有田隆也，牛嶋昌和，I. Rittaporn 著，日本ソフトバンク刊

9．I/O '85年7月号「マイクロプロセッサを比較する⑲」，Processor's Professor 著，工学社刊

10．「最新フロッピ・ディスク装置とその応用ノウハウ」，高橋昇司著，CQ出版社刊

11．「ガラスの仮面」1～34巻，美内すずえ著，白泉社刊

12．「小さなお茶会」1～6巻，猫十字社著，白泉社刊

# 索　引

## I



## J



## K



## L



## M



## N



## O



## P



## R



## S



## T

《著者略歴》

```
E000 24 45 4D 06 24 31 53 AE : 12
E008 C1 B3 C9 FF 23 FF F0 01 : 4F
E010 76 FF 79 ED E0 38 01 23 : 17
E018 E0 2C CD FF E0 38 11 23 : 24
E020 FF 23 23 FF 00 AB 01 23 : 13
E028 FF 01 10 01 E0 32 11 09 : 3D
E030 00 12 01 FF 2A 3E 23 AB : 48
E038 01 07 A8 21             : D1
---------------------------------
SUM: 3A 60 38 11 11 BB 8A CC D5B4
```

# 試験に出るX1

ハードウェアのフルコース

定価はカバーに記載されております

昭和62年12月15日　初版発行
昭和63年6月15日　第4刷発行

著　者　祝　一平

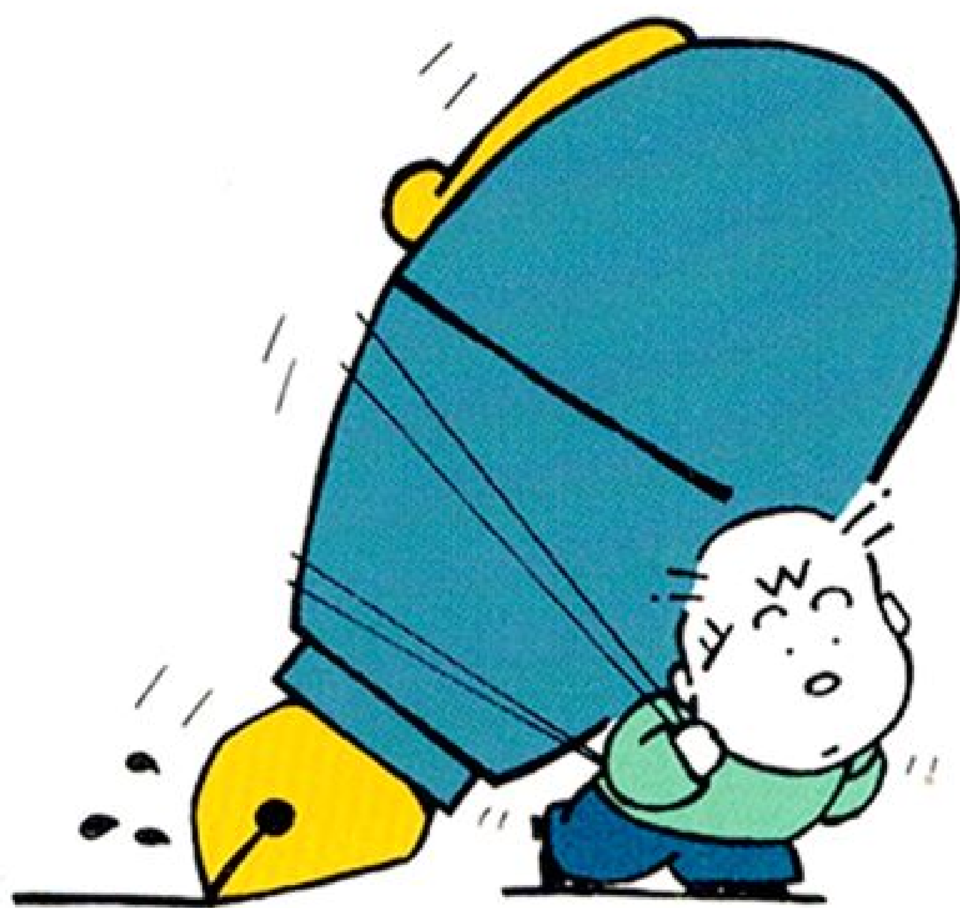
発行者　孫　正義
発行所　株式会社日本ソフトバンク
出版事業部
〒102　東京都千代田区九段南2-3-26
☎03(261)4095
印　刷　壮光舎印刷株式会社



ISBN4-930795-88-5


パソコンテレビ
X1
turbo
SHARP

SOFT BANK



ISBN4-930795-88-5 C0055 ¥2800E

定価 2,800円